Εργαστηριακή Άσκηση Μέρος Α’ 2020-2021.

Ανδρούτσος Λάμπρος

Α.Μ : 1054396

Η εργασία αυτή δημιουργήθηκε στα πλαίσια του μαθήματος :

Υπολογιστική Νοημοσύνη

Τμήμα Μηχανικών Ηλεκτρονικών Υπολογιστών και Πληροφορικής

Πανεπιστήμιο Πατρών

Ελλάδα

20/04/2021

Περιεχόμενα

Link για τον κώδικα: https://github.com/labrosandroutsos/ANN-CNN-assignment

Α1. Προεπεξεργασία και Προετοιμασία δεδομένων

α) Επιλογή μεθόδου προσαρμογής των τιμών του dataset σε διαφορετική κλίμακα

Γενικά, στην μηχανική μάθηση υπάρχουν τρεις συχνοί τρόποι προσαρμογής των τιμών ενός dataset**, Κεντράρισμα, Κανονικοποίηση** και **Τυποποίηση**, γνωστά με τα αγγλικά τους ονόματα, **Centering, Normalization** και **Standardization**.

Θα εξηγήσουμε συνοπτικά τον σκοπό κάθε μεθόδου.

Centering: Διαδικασία αφαίρεσης του μέσου όρου της μεταβλητής από κάθε σημείο του dataset ώστε ο μέσος όρος της μεταβλητής να γίνει 0. Στο συγκεκριμένο πρότζεκτ ασχολούμαστε με δεδομένα από pixels εικόνας, οπότε το centering αφαιρεί τον μέσο όρο των τιμών των pixels της εικόνας από όλες τις τιμές.

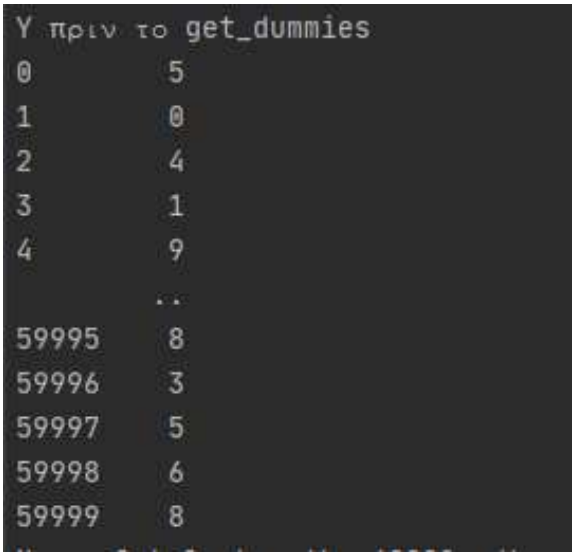
Normalization: Διαδικασία αλλαγής κλίμακας των δειγμάτων ώστε όλα να έχουν ομοιόμορφη κλίμακα. Ουσιαστικά, θα μεταφέρω όλες τις τιμές των pixels στην κλίμακα [0,1]. Επιλέγουμε το εύρος τιμών [0,1] γιατί είναι πολύ σύνηθες και καλή πρακτική να χρησιμοποιείται αυτό το εύρος για Νευρωνικά Δίκτυα.

Standardization: Διαδικασία αλλαγής κλίμακας των δειγμάτων ώστε η μέση τιμή και η τυπική απόκλιση τους να είναι 0 και 1, αντίστοιχα.

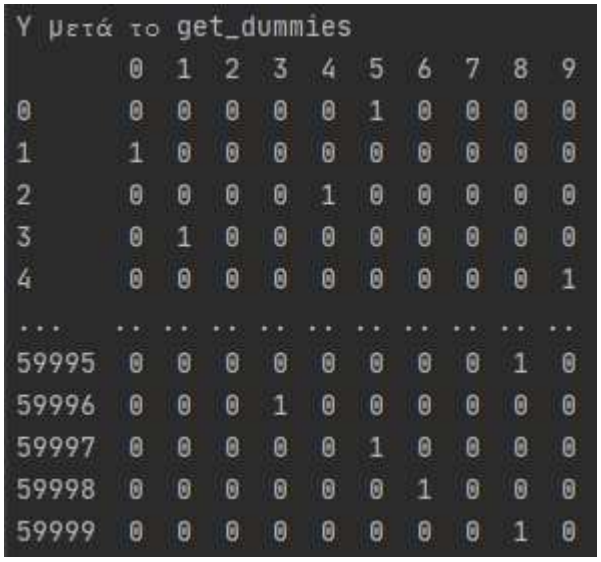
Με βάση τον τύπο των δεδομένων του dataset που μας δίνεται, θεώρησα πως ο καλύτερος τρόπος προσαρμογής των τιμών τους θα ήταν το Normalization, καθώς όλες οι τιμές είναι στο εύρος 0 έως 255, οπότε με το Normalization θα μεταφερθούν στο εύρος 0 έως 1 που είναι ιδανικό για το νευρωνικό δίκτυο που θα φτιάξουμε. Οπότε θα χρησιμοποιηθεί η συνάρτηση **MinMaxScaler** της βιβλιοθήκης scikit-learn, που ως default έχει την κανονικοποίηση των στοιχείων στο [0, 1].

Αξίζει να αναφερθεί η μετατροπή των τιμών των pixels στα train και test datasets από unsigned ints σε floats, ώστε να γίνει μετά η προσαρμογή των τιμών μέσω των παραπάνω τριών μεθόδων.

Επίσης, χώρισα το dataset αυτά σε X και y, όπου y είναι τα labeled data (categorical χαρακτηριστικό το label column του dataset) μας ενώ X τα input data μας (όλες οι στήλες χωρίς το label). Για το y χρησιμοποίησα και την συνάρτηση get_dummies() της βιβλιοθήκης pandas που χρησιμοποιείται για ανάλυση δεδομένων. Η συνάρτηση αυτή μετατρέπει την κατηγορική μεταβλητή μας σε dummy μεταβλητές.



Εικόνα 1| Y_train πριν το get_dummies



Εικόνα 2| Y_train μετά το get_dummies

Όπως παρατηρούμε από τις δύο παραπάνω εικόνες, στην αρχή το y είχε διαστάσεις 60000x1 ενώ μετά την εφαρμογή του pd.get_dummies() πλέον έχει διαστάσεις 60000x10, αφού 10 είναι οι αριθμοί που πρέπει να προβλέπει το μοντέλο μας, Κάθε γραμμή του μητρώου αποτελεί ένα δείγμα και το δείγμα αυτό έχει 1 μόνο στην κλάση στην οποία αντιστοιχεί, πχ η πρώτη γραμμή του πίνακα ανήκει στην κλάση 5.

Ουσιαστικά δημιουργήσαμε μία δυαδικοποίηση (binarization) του κατηγορικού χαρακτηριστικού label, 1 σημαίνει ότι ανήκει σε αυτήν την κλάση, ενώ 0 ότι δεν ανήκει.

β) Διασταυρούμενη επικύρωση

Θα χωρίσουμε τα δεδομένα μας σε 5 υποσύνολα (5-fold Cross Validation). Τα υποσύνολα αυτά θα περιέχουν διαφορετικές παρατηρήσεις από το dataset μας. Τα 4 από αυτά τα υποσύνολα θα χρησιμοποιηθούν για το σύνολο εκπαίδευσης, ενώ το 5^{ο} θα χρησιμοποιηθεί ως σύνολο επικύρωσης (validation set). Το μοντέλο που θα φτιάξουμε σε επόμενη παράγραφο της εργασίας θα εκπαιδεύεται κάθε φορά με το σύνολο εκπαίδευσης και ένα διαφορετικό σύνολο επικύρωσης κάθε φορά, δηλαδή αυτό επαναλαμβάνεται 5 φορές ώστε όλα τα υποσύνολα να γίνουν μία φορά σύνολο επικύρωσης. Μετά από αυτές τις 5 επαναλήψεις υπολογίζουμε τη μέση απόδοση του μοντέλου. Ουσιαστικά, η τεχνική αυτή μας βοηθάει στο να εκτιμήσουμε την απόδοση του μοντέλου μας με νέα δεδομένα testing, δηλαδή την ικανότητα του μοντέλου μας να γενικεύει.

Α2. Επιλογή αρχιτεκτονικής

α) Εκπαίδευση και αξιολόγηση των μοντέλων με χρήση Cross-Entropy και Μέσου Τετραγωνικού Σφάλματος (MSE)

Αρχικά, αναφέρουμε πως το συγκεκριμένο πρόβλημα αποτελείται από 10 κλάσεις, που αντιπροσωπεύουν τους αριθμούς 0 έως 9, άρα μιλάμε για πρόβλημα πολλαπλών κλάσεων (πάνω από δύο κλάσεις).

Η πιο συχνή συνάρτηση κόστους για προβλήματα παλινδρόμησης (**Regression**) είναι το **Μέσο Τετραγωνικό Σφάλμα (MSE)** που χρησιμοποιείται στην πρόβλεψη αριθμητικών τιμών, δηλαδή το αποτέλεσμα θα είναι ένας αριθμός και αποτελεί τη μέση τετραγωνική διαφορά μεταξύ των προβλεπόμενων τιμών και των πραγματικών τιμών και είναι πάντα θετικό, ενώ το cross entropy μπορεί να είναι και αρνητικό. Με αυτή τη συνάρτηση κόστους θα θέλαμε οι έξοδοι να αποτελούν μία συνάρτηση πραγματικών τιμών των εισόδους. Το MSE είναι γενικά αρκετά καλή συνάρτηση κόστους όταν γνωρίζουμε ότι τα δεδομένα μας είναι ομοιόμορφα κατανεμημένα γύρω από μία μέση τιμή και θέλουμε να εφαρμόσουμε μεγάλο error σε outliers.

$$L(y, \hat{y}) = \frac{1}{N} \sum_{i=0}^{N} (y - \hat{y}_i)^2$$

Εικόνα 3| MSE loss function

Για Cross-Entropy θα χρησιμοποιήσουμε **Categorical Cross-Entropy** καθώς έχουμε ένα πρόβλημα πολλαπλών κλάσεων, όπου η έξοδος της θα είναι ένα διάνυσμα τιμών, με τιμή 1 στην κλάση στην οποία ανήκει η είσοδος και 0 για τις κλάσεις στις οποίες δεν ανήκει η είσοδος. Για την συγκεκριμένη μετρική πρέπει τα labels να είναι σε μορφή αντίστοιχη που προκύπτει από το pd.get_dummies() (**One_hot representation**). Αν θέλαμε να εισάγουμε τα labels σαν αριθμούς, θα χρησιμοποιούσαμε **sparse Categorical Cross-Entropy**.

$$C = -\frac{1}{n} \sum_{x} [y * lna + (1 - y) * \ln(1 - a)]$$

Εικόνα 4| CE loss function

Προφανώς και στις δύο μετρικές θέλουμε όσο το δυνατόν μικρότερες τιμές!

β) Είσοδοι στο ΤΝΔ

Εφόσον μία είσοδος πρέπει να αναπαριστά 28x28 εικόνα, άρα μία εικόνα με 784 pixels. Θεωρώ ότι θα χρειαστούμε **784 εισόδους**, μία για κάθε pixel της εικόνας, δηλαδή input_shape=784, αφού τα δεδομένα μας έχουν γίνει Flatten.

γ) Νευρώνες στο επίπεδο εξόδου

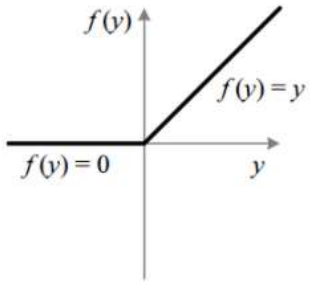
Θα χρειαστούμε **10 νευρώνες** στο επίπεδο εξόδου γιατί τόσες είναι οι διαφορετικές και μοναδικές κλάσεις για το πρόβλημα αυτό και είναι οι αριθμοί από το 0 έως το 9.

δ) Συνάρτηση ενεργοποίησης στους κρυφούς κόμβους

Ως συνάρτηση ενεργοποίησης στο κρυφό επίπεδο θα χρησιμοποιήσουμε την **ReLU**. Αρχικά θα αναλύσουμε την ReLU και τον λόγο που την επιλέγουμε και δεν επιλέγουμε μία παραλλαγή της.

Η **ReLU** αποτελείται ίσως την πιο συχνή (τα τελευταία χρόνια) συνάρτηση ενεργοποίησης για τα κρυφά επίπεδα. Για την ακρίβεια, έχει γίνει η default συνάρτηση ενεργοποίησης για τα επίπεδα αυτά. Αποτελεί μία απλή και αποτελεσματική συνάρτηση η οποία είναι λιγότερο επιρρεπής στο φαινόμενο εξαφανιζόμενων κλίσεων (vanishing gradient problem) που αποτρέπει τα βαθιά νευρωνικά μοντέλα να εκπαιδευτούν. Η συνάρτηση της ReLU είναι: **max(0, x)**, δηλαδή μηδενίζει όλες τις αρνητικές τιμές, ενώ αφήνει όπως έχουν όλες τις θετικές τιμές. Ακολουθεί η γραφική της παράσταση:

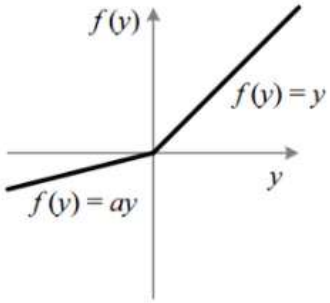
:



Εικόνα 5| ReLU γραφική παράσταση

Η συνάρτηση αυτή συγκλίνει γρήγορα, αλλά έχει ένα πρόβλημα που ονομάζεται Dying ReLU. Ουσιαστικά, όταν ένας νευρώνας γίνεται αρνητικός παίρνει τιμή 0 μέσω της ReLU και μετά μπορεί να μην ανακάμψει ποτέ και άρα τέτοιοι νευρώνες δεν θα παίζουν κανένα ρόλο στο πρόβλημα μας, δεν θα εκπαιδεύονται και δεν θα χρησιμοποιούνται τελικά. Για τον λόγο αυτό θα χρησιμοποιήσουμε την LeakyReLU.

Η διαφορά της LeakyReLU από την κανονική είναι ότι δεν μηδενίζει τις αρνητικές τιμές, αλλά τους δίνει μία πολύ μικρή θετική τιμή. Η συνάρτηση της είναι: max(ax,x), a στο διάστημα [0,1], πχ a=0.01*y. Η συνάρτηση αυτή είναι και πιο γρήγορη στην φάση της εκπαίδευσης από την κανονική ReLU και είναι πιο ισορροπημένη.



Εικόνα 6| LeakyReLU γραφική παράσταση

Στην αρχή δοκιμάστηκε η LeakyReLU με παράμετρο 0.01, όμως παρουσίασε χειρότερα αποτελέσματα από την κανονική ReLU. Η LeakyReLU εισάγει μία επιπλέον παράμετρο στο δίκτυο, η οποία χρειάζεται βελτιστοποίηση και δεν είναι

αποδοτικό να ασχοληθούμε με αυτήν. Για τους λόγους αυτούς θα παραμείνουμε στην ReLU που όπως φαίνεται παράγει και καλύτερα αποτελέσματα στις μετρικές μας.

ε) Συνάρτηση ενεργοποίησης στο επίπεδο εξόδου

Εφόσον ασχολούμαστε με πρόβλημα πολλαπλών κλάσεων, θα χρησιμοποιήσουμε την συνάρτηση ενεργοποίησης **softmax** στο επίπεδο εξόδου. Αυτή η συνάρτηση αποτελεί γενίκευση της λογιστικής συνάρτησης θα επιστρέψει έναν πίνακα με 10 πιθανοτικά σκορ, όπου κάθε σκορ θα αντιπροσωπεύει την πιθανότητα το δείγμα αυτό να ανήκει σε μία από τις 10 κλάσεις μας. Έτσι κάθε τιμή εξόδου θα αποτελεί μία κατηγοριματική συνάρτηση πιθανότητα. Η συνάρτηση της είναι: $\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$

στ) Πειράματα με 3 διαφορετικές τιμές νευρώνων στο κρυφό επίπεδο

Αφού το πρόβλημα μας είναι ένα πρόβλημα πολλαπλών κλάσεων, περιμένουμε πως η συνάρτηση κόστους **Categorical Cross-Entropy** θα έχει τα καλύτερα αποτελέσματα στην ακρίβεια του μοντέλου. Θα μπορούσαμε να χρησιμοποιήσουμε και την Sparse Categorical Cross-Entropy, αλλά αυτή χρησιμοποιείται για δεδομένα που δεν έχουν την μορφή one hot encoding.

Θα χρησιμοποιήσουμε **100 εποχές εκπαίδευσης** για αρχή και δεν θα χρησιμοποιήσουμε kernel και bias initializer, καθώς στο σημείο αυτό δεν μας ενδιαφέρει πως θα οριστούν τα τυχαία βάρη στα επίπεδα του μοντέλου μας.

Optimizer: θα χρησιμοποιήσουμε τον **SGD**, καθώς έχει και παράμετρο σταθεράς ορμής (momentum) που ζητείται σε επόμενο ερώτημα, και παράμετρο μάθησης 0.001 όπως μας δίνεται στην εκφώνηση της άσκησης. Ο SGD χρησιμοποιεί ένα υποσύνολο του συνόλου εκπαίδευσης ώστε να υπολογίσει την κλίση της συνάρτησης κόστους. Η κλίση αυτή είναι μια στοχαστική προσέγγιση της πραγματικής κλίσης του κόστους και για αυτό ο optimizer αυτός μειώνει την πιθανότητα εγκλωβισμού της μάθησης σε ένα τοπικό ελάχιστο.

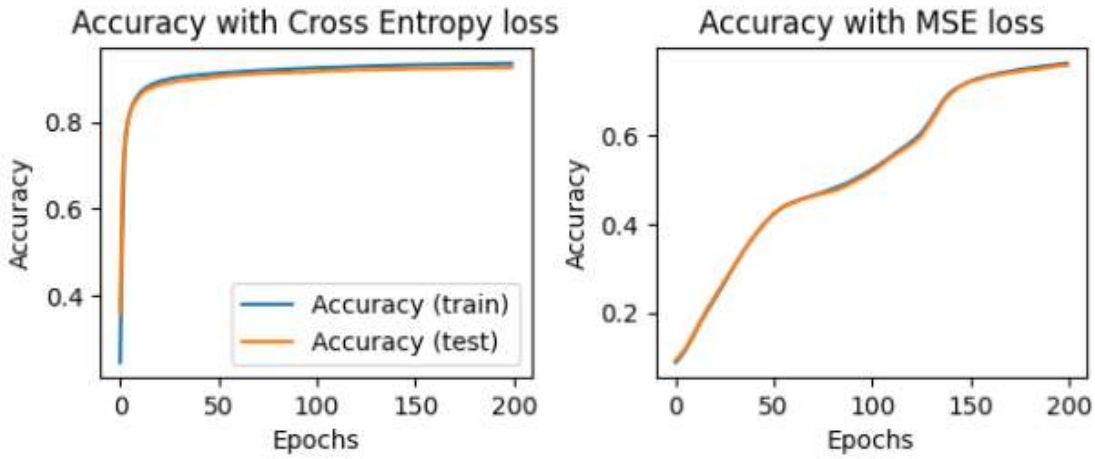
Batch size: Αν χρησιμοποιήσουμε ένα αρκετά μεγάλο batch size, είναι πολύ πιθανό να οδηγηθεί το μοντέλο σε κακή δυνατότητα γενίκευσης. Από την άλλη μεριά, αν χρησιμοποιήσουμε ένα μικρό batch size, τότε δεν είναι σίγουρο ότι το μοντέλο μας θα συγκλίνει στο ολικό μέγιστο (**global maxima**). Ως καλή πρακτική θεωρείται να ξεκινάει κάποιος με ένα μικρό batch size και σταδιακά να το αυξάνει μέχρι να βρει το ιδανικό για το πρόβλημα του, ώστε να υπάρχει και σίγουρη σύγκλιση αλλά και γρήγορη εκπαίδευση του μοντέλου. Στην αρχή θα χρησιμοποιούσαμε 256 batch size, που είναι σχετικά μικρό για τον όγκο των δεδομένων που έχουμε (48000 samples στο X_train), αλλά μετά από το πείραμα αυτό ξανατρέξαμε τα μοντέλα για 32 batch size που αποτελεί και το default batch size του Keras module, και τελικά τα αποτελέσματα ήταν καλύτερα, οπότε θα χρησιμοποιηθεί 32 batch size (αφού το 32 διαιρεί και τέλεια το 48000 που αποτελεί τον αριθμό των samples στο X_train) ακόμα και αν επιβραδύνει αρκετά τον χρόνο εκτέλεσης του κώδικα.

Δεν θα χρησιμοποιήσουμε κάποιο κριτήριο τερματισμού της εκπαίδευσης στο ερώτημα αυτό. Θα το μελετήσουμε αυτό σε επόμενο ερώτημα με το βέλτιστο μοντέλο μας.

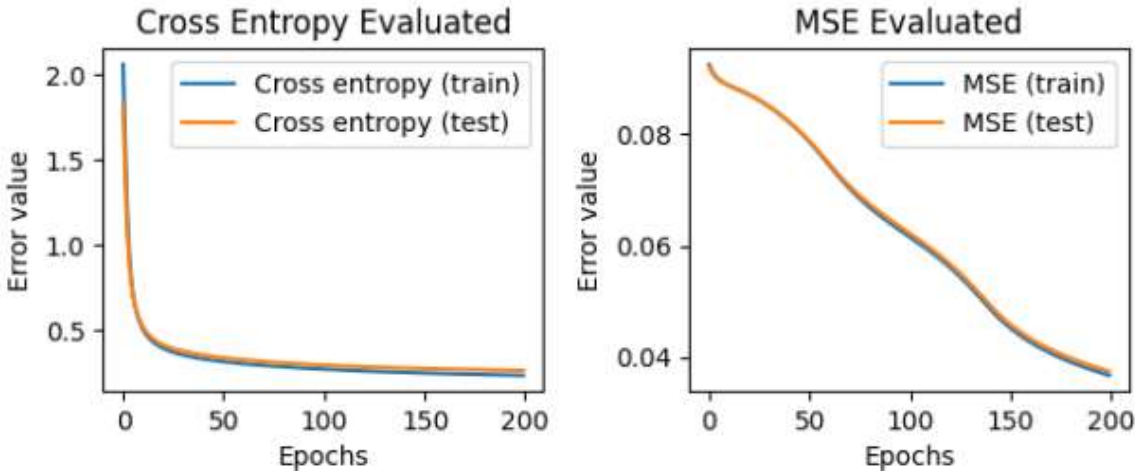
Οι τιμές των μετρικών στον παρακάτω πίνακα προέκυψαν από την μέση τιμή των 5 τιμών τους (από **το 5fold cross validation**). Επίσης, κάθε φορά που τρέχει ο κώδικας, εκπαιδεύει 2 νευρωνικά δίκτυα, ένα με **CE loss** και ένα με **MSE loss** για να μειωθούν συνολικά οι φορές που πρέπει να τρέξουμε τον κώδικα. Επίσης, σε κάθε fold χρησιμοποιήθηκε ως validation data τα test datasets που προκύπτουν από τον διαχωρισμό σε 5 folds, ώστε να μπορούμε να δούμε την γενικευτική ικανότητα του μοντέλου σε δεδομένα στα οποία δεν έχει εκπαιδευτεί. Μετά, στο evaluate του μοντέλου χρησιμοποιήθηκε το αρχικό mnist test dataset ώστε να δούμε τα αποτελέσματα για τις συναρτήσεις κόστους και την ακρίβεια στο mnist test, που δεν έχει πάρει μέρος στην εκπαίδευση του μοντέλου. Έτσι, μπορούμε να έχουμε μια πλήρης εικόνα για την γενικευτική ικανότητα του μοντέλου και την απόδοση του.

Αριθμός νευρώνων στο κρυφό επίπεδο	**CE loss**	**MSE**
H1 = O = 10	0.2565726	0.0358113
H1=(I+O)/2=784+10/2=397	0.1195953	0.0166156
H1 = I+O = 794	0.1162988	0.0161631

Για 10 νευρώνες στο κρυφό επίπεδο:

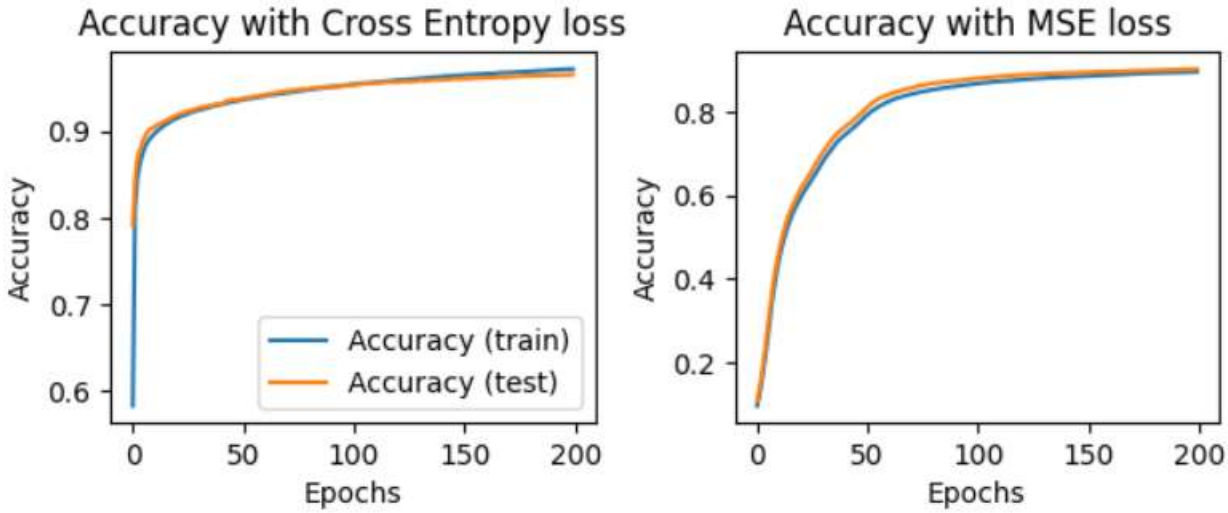


Εικόνα 7| Ακρίβεια για 10 νευρώνες στο 1ο κρυφό επίπεδο

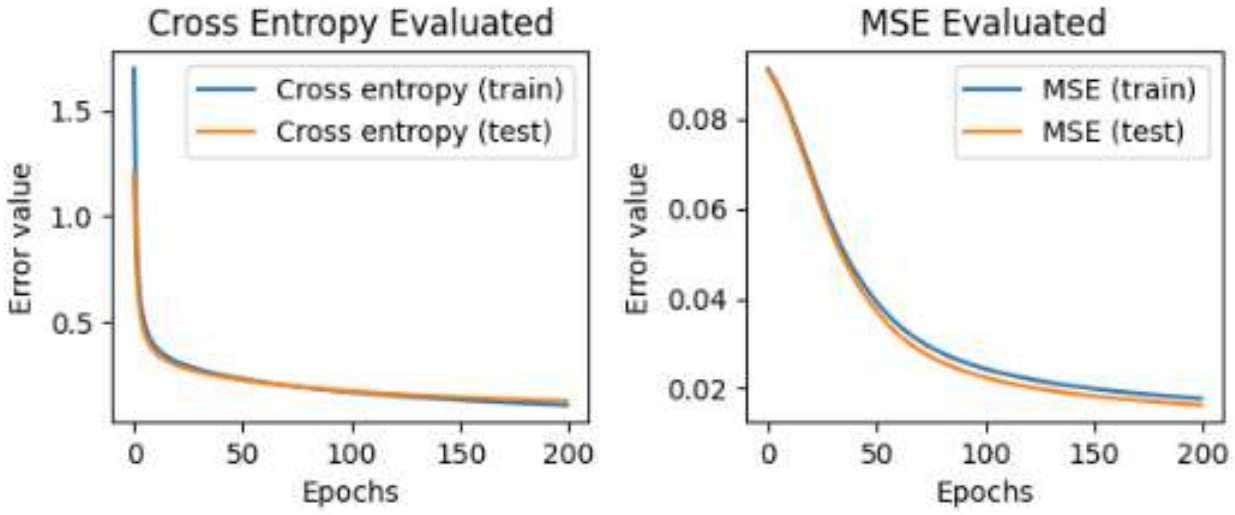


Εικόνα 8| CE και MSE loss για 10 νευρώνες στο 1ο κρυφό επίπεδο

Για 397 νευρώνες στο κρυφό επίπεδο:

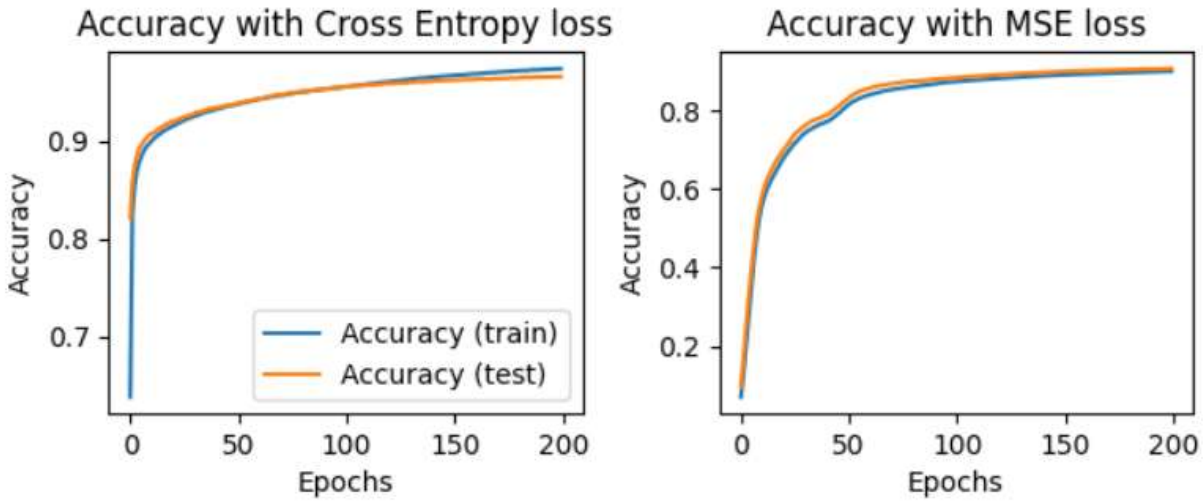


Εικόνα 9| Ακρίβεια για 397 νευρώνες στο 1ο κρυφό επίπεδο

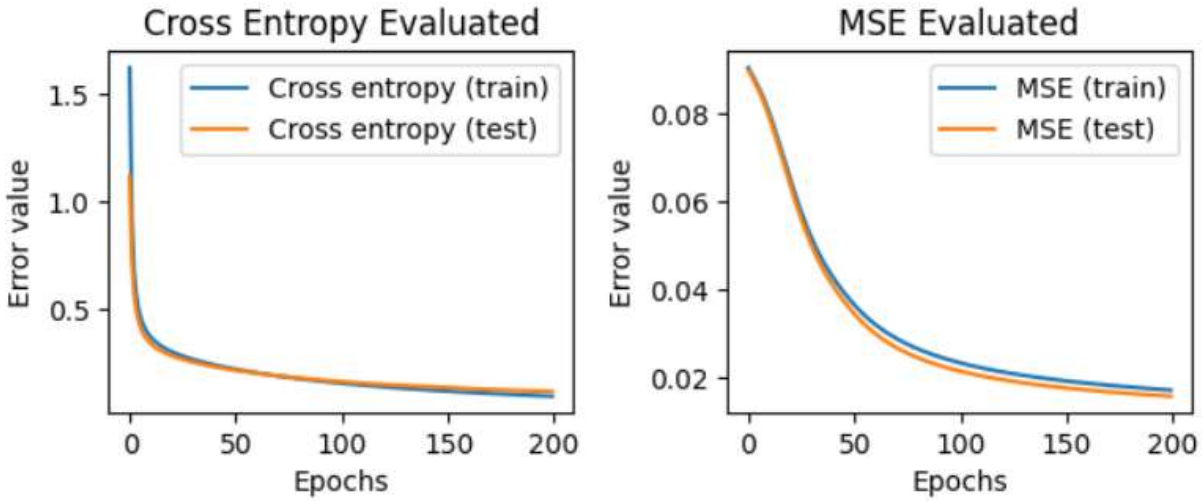


Εικόνα 10| CE και MSE loss για 397 νευρώνες στο 1ο κρυφό επίπεδο

Για 794 νευρώνες στο κρυφό επίπεδο:



Εικόνα 11| Ακρίβεια για 794 νευρώνες στο 1ο κρυφό επίπεδο



Εικόνα 12|CE και MSE loss για 794 νευρώνες στο 1ο κρυφό επίπεδο

Το μοντέλο μας δεν υπερεκπαιδεύεται για καμία από τις 3 επιλογές νευρώνων για το κρυφό επίπεδο.

Από τις παραπάνω γραφικές παρατηρώ αρχικά πως το μοντέλο μαθαίνει να κατηγοριοποιεί τα δεδομένα αρκετά γρήγορα σε λίγες εποχές. άρα θα επωφεληθεί σίγουρα το μοντέλο μας από κάποιο κριτήριου τερματισμού, ώστε να μην χρειάζεται να εκπαιδευτεί και για τις 200 εποχές.

Όσον αφορά τον αριθμό των κρυφών κόμβων, παρατηρούμε ότι με 10 νευρώνες στο κρυφό επίπεδο έχουμε τα χειρότερα αποτελέσματα με τις συναρτήσεις κόστους και στην ακρίβεια του μοντέλου, αλλά και στην ταχύτητα σύγκλισης. Όμως, με 397 και 794 νευρώνες στο κρυφό επίπεδο επιτυγχάνει καλύτερα αποτελέσματα στις συναρτήσεις κόστους Cross Entropy και MSE, με τους 794 νευρώνες να είναι απειροελάχιστα καλύτεροι, με την ίδια όμως ακρίβεια (γύρω στο 96.5% για train και

test). Παρ'όλ'αυτά, με 794 νευρώνες το μοντέλο χρειάζεται πολύ περισσότερη ώρα για να εκπαιδευτεί, ενώ δεν προσφέρει τελικά περισσότερη ακρίβεια στο πρόβλημα μας. Οπότε, είναι προτιμότεροι οι 397 νευρώνες που χρειάζονται και λιγότερο υπολογιστικό κόστος. **Για τον λόγο αυτό θα επιλέξουμε τους 397 νευρώνες για το 1ο κρυφό επίπεδο.**

Η Categorical Cross Entropy σαν συνάρτηση κόστους φαίνεται πως δουλεύει αρκετά καλύτερα με το πρόβλημα μας σε αντίθεση με την MSE, καθώς παρατηρείται καλύτερη ακρίβεια (accuracy) με την χρήση της Cross entropy loss.(97.2% έναντι 90.5% με την MSE για 397 και 794 νευρώνες, και ~93% έναντι ~77 % για 10 νευρώνες)

Όπως έχει αναφερθεί ήδη, παρατηρείται πως το μοντέλο μας φτάνει αρκετά γρήγορα στην σύγκλιση για το πρόβλημα αυτό και για τις 3 διαφορετικές τιμές νευρώνων στο κρυφό επίπεδο, όμως για 10 νευρώνες είναι πιο αργή η σύγκλιση ειδικά με MSE συνάρτηση κόστους. Αυτό παρατηρείται **με Cross Entropy** σαν loss. Με **MSE** loss έχουμε πιο αργή σύγκλιση. Αυτό παρατηρείται από όλες τις παραπάνω γραφικές παραστάσεις. Το αποτέλεσμα αυτό ήταν αναμενόμενο. Αρχικά, δίνοντας πολλές εποχές σε μία εκπαίδευση, είναι πιο πιθανό κάποια στιγμή να καταλήξει σε σύγκλιση, εφόσον δεν υπερεκπαιδεύεται. Στο πρόβλημα μας ,όπως έχει ήδη αναφερθεί, είναι καταλληλότερη η Categorical Cross Entropy σαν συνάρτηση κόστους και αυτό αποδεικνύεται και από τις γραφικές παραστάσεις, όμως η MSE όντας μία πολύ καλή συνάρτηση κόστους και με τα δεδομένα μας κανονικοποιημένα στο [0, 1] καταλήγει και αυτή σε σύγκλιση το πρόβλημα, αλλά σε πιο πολλές εποχές.

Τελικά, προτιμάται η **Categorical Cross Entropy** σαν συνάρτηση κόστους, καθώς καταλήγει σε καλύτερη ακρίβεια του μοντέλου μας και πιο γρήγορη σύγκλιση.

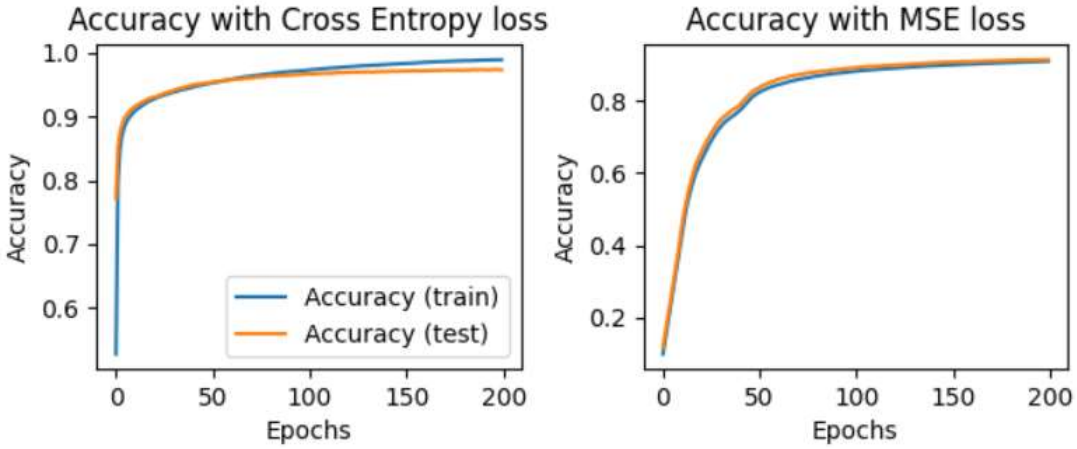
ζ) Πρόσθεση ενός κρυφού επιπέδου ακόμα

Θα δοκιμάσουμε πέντε τιμές, ίδιο αριθμό νευρώνων με το πρώτο κρυφό επίπεδο, τους μισούς νευρώνες σε σχέση με το πρώτο κρυφό επίπεδο, διπλάσιους νευρώνες σε σχέση με το πρώτο κρυφό επίπεδο και δύο τιμές μικρότερές του αριθμού των νευρώνων του πρώτου κρυφού επιπέδου, αλλά διαφορετική από την δεύτερη τιμή.

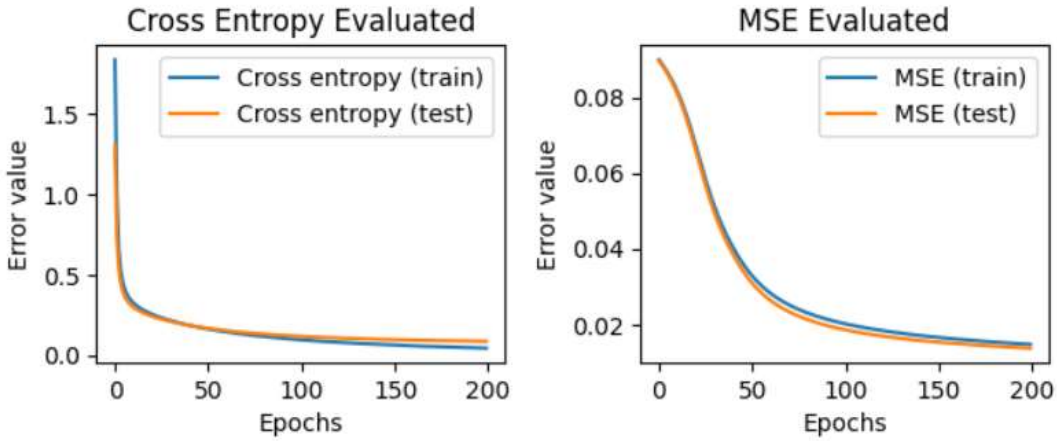
Αριθμός νευρώνων στο νέο κρυφό επίπεδο	**CE loss**	**MSE**
H1 = 100	0.08459163	0.01419769
H2 = 397	0.08483210	0.01436415
H3 = 600	0.08628921	0.01436316

Για 100 νευρώνες στο 2ο κρυφό επίπεδο:

(97.4% ακρίβεια έναντι 91.2%)



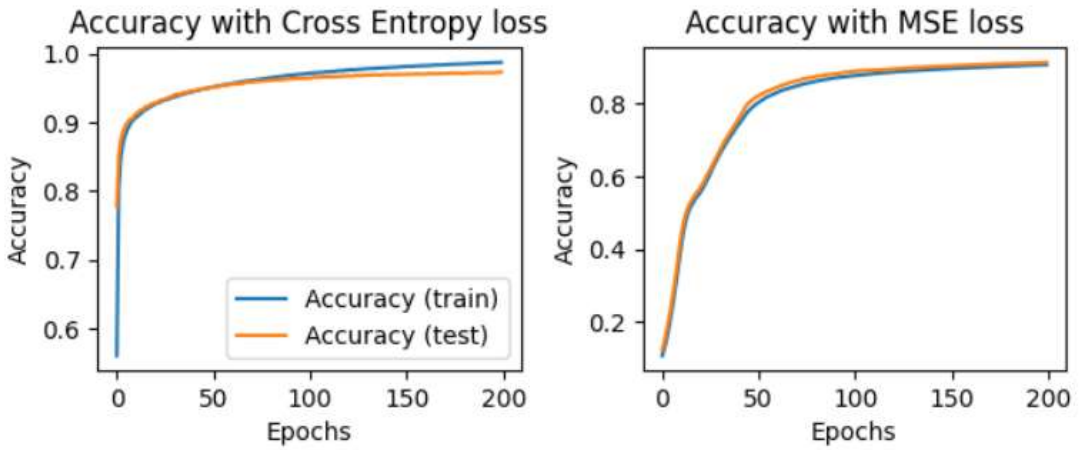
Εικόνα 13| Ακρίβεια για 100 νευρώνες στο 2ο κρυφό επίπεδο



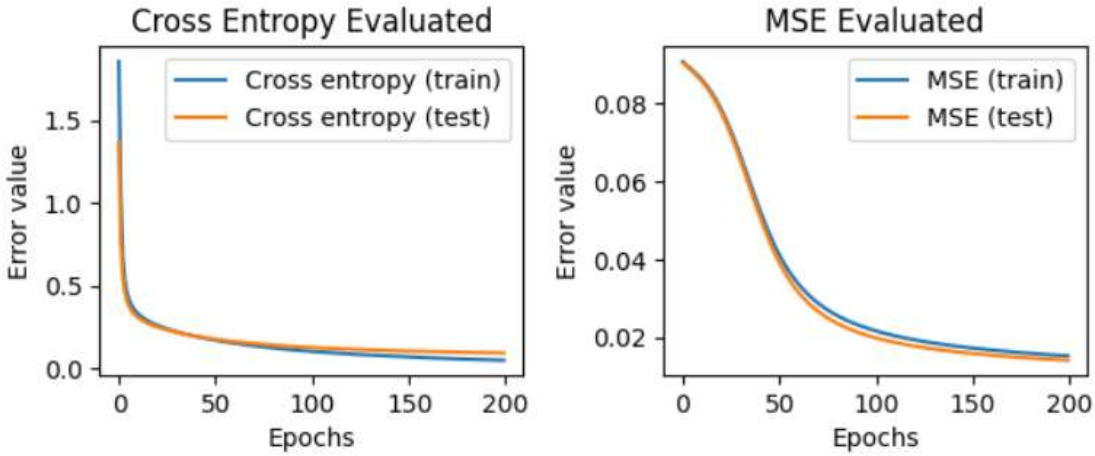
Εικόνα 14| CE και MSE loss για 100 νευρώνες στο 2ο κρυφό επίπεδο

Για 397 νευρώνες στο 2ο κρυφό επίπεδο:

(98% ακρίβεια έναντι 91.9%)



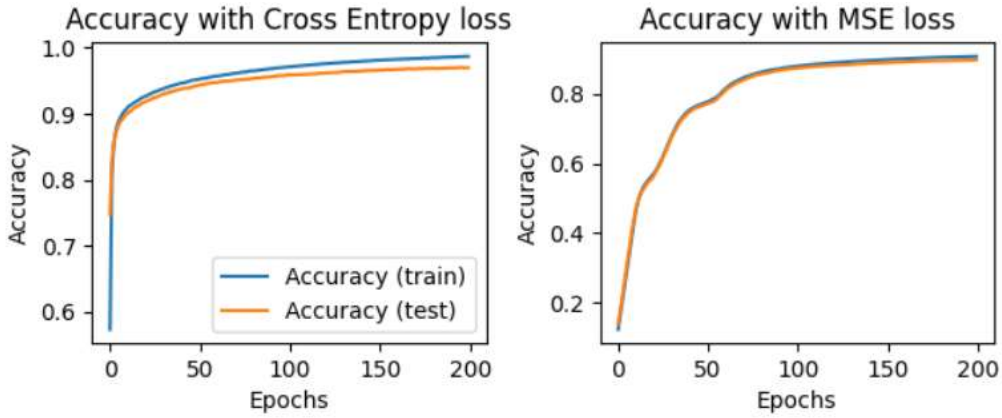
Εικόνα 15| Ακρίβεια για 397 νευρώνες στο 2ο κρυφό επίπεδο



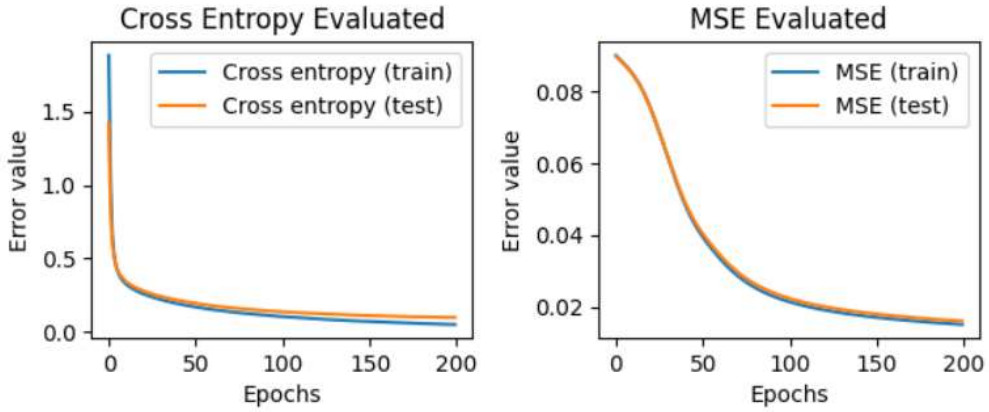
Εικόνα 16| CE και MSE loss για 397 νευρώνες στο 2ο κρυφό επίπεδο

Για 600 νευρώνες στο 2ο κρυφό επίπεδο:

(98.5% ακρίβεια έναντι 92.3%)



Εικόνα 17| Ακρίβεια για 600 νευρώνες στο 2ο κρυφό επίπεδο



Εικόνα 18| CE και MSE loss για 600 νευρώνες στο 2ο κρυφό επίπεδο

Σύγκριση μεταξύ ενός και δύο κρυφών επιπέδων:

Αριθμός κρυφών επιπέδων	**CE loss**	**MSE**
1 κρυφό επίπεδο με 397 νευρώνες	0.1195953	0.0166156
2 κρυφά επίπεδα με 397 και 100 νευρώνες	0.0845916	0.0141976

Πλέον έχουμε πιο βαθιά νευρωνικά. Προσθέτουμε ένα επιπλέον επίπεδο αυξάνοντας αρκετά τον αριθμό των πράξεων και την πολυπλοκότητα του μοντέλου. Αυτό έχει ως αποτέλεσμα την αύξηση της ακρίβειας και καλύτερα αποτελέσματα στις συναρτήσεις κόστους όπως φαίνεται και από τις παρακάτω γραφικές παραστάσεις, αλλά προσθέτει μεγάλο υπολογιστικό κόστος και πολύ μεγαλύτερους χρόνους εκτέλεσης (πολύ περισσότερες πράξεις).

Και οι 3 διαφορετικές τιμές νευρώνων στο 2ο κρυφό επίπεδο, προσφέρουν καλύτερη ακρίβεια και αποτελέσματα στις συναρτήσεις κόστους σε σχέση με το καλύτερο μοντέλο του προηγούμενου ερωτήματος με 397 νευρώνες σε 1 κρυφό επίπεδο μόνο.

Όσο αυξάνονται οι νευρώνες στο 2ο κρυφό επίπεδο, φαίνεται πως ανεβαίνει το ποσοστό της ακρίβειας και τα αποτελέσματα στις συναρτήσεις κόστους είναι σχεδόν ίδια. Αν και αυτό δείχνει ότι με περισσότερους νευρώνες στο 2ο κρυφό επίπεδο, έχουμε καλύτερο μοντέλο για το συγκεκριμένο πρόβλημα, το υπολογιστικό κόστος και η αύξηση του χρόνου εκτέλεσης δεν είναι αντίστοιχη της αύξησης στην ακρίβεια. Επίσης, στην βιβλιογραφία δεν υπάρχει σαφής απάντηση για τον αριθμό των κόμβων σε διαδοχικά κρυφά επίπεδα (ίδιο, μειούμενο, αυξανόμενο). Συνήθως, η επιλογή του αριθμού των κόμβων προκύπτει από πειραματισμό, καθώς ο αριθμός των κόμβων αποτελεί υπερπαράμετρος των νευρωνικών δικτύων.

Στο συγκεκριμένο πρόβλημα με βάση τα αποτελέσματα στην ακρίβεια είναι προτιμότερο να έχουμε αυξανόμενο αριθμό κόμβων στο 2ο κρυφό επίπεδο.

Στο συγκεκριμένο πρότζεκτ όμως θα επιλέξουμε την χρήση ενός κρυφού επιπέδου, καθώς επιτυγχάνει 97% ακρίβεια σε πολύ λιγότερο χρόνο και με μικρότερο υπολογιστικό κόστος. Το μοντέλο αυτό καταφέρνει πολύ καλά να ξεχωρίζει τις κλάσεις και να γενικεύει, άρα θα παραμείνουμε σε αυτό για τα επόμενα ερωτήματα.

η) Κριτήριο τερματισμού

Ένας από τους πιο συνηθισμένους τρόπους αποφυγής της υπερεκπαίδευσης ενός μοντέλου είναι η χρήση κάποιου κριτήριου τερματισμού της εκπαίδευσης. Εγώ θα χρησιμοποιήσω το **πρόωρο σταμάτημα** ή αλλιώς **Early Stopping**, η οποία αποτελεί μορφή ομαλοποίησης.

Συνήθως μετά από κάποιον αριθμό εποχών εκπαίδευσης ενός μοντέλου το σφάλμα επικύρωσης (validation set error) αρχίζει να αυξάνεται, το οποίο αποτελεί σημάδι υπερεκπαίδευσης. Σε αυτό το σημείο χρησιμοποιείται το Early Stopping ώστε να σταματάει η εκπαίδευση για κάθε fold όταν παρατηρηθεί πτώση της απόδοσης του μοντέλου στο σύνολο δεδομένων επαλήθευσης. Παρ'όλ'αυτά στο πρόβλημα μας δεν παρατηρείται overfitting καθώς τα train και test accuracies είναι αρκετά κοντά μεταξύ τους, είτε με MSE είτε με CE loss. Το Early Stopping ίσως βοηθήσει στο πρόβλημα αυτό καθώς όπως φαίνεται από τις γραφικές παραστάσεις, δεν χρειάζονται και οι 200 εποχές για την εκπαίδευση του μοντέλου. Μπορούμε να επιτύχουμε πολύ καλές τιμές με λιγότερες εποχές και το προτιμάμε για το συγκεκριμένο πρότζεκτ/πρόβλημα. Θα τρέξουμε τα νευρωνικά μας με EarlyStopping callback και θα καταλήξουμε στο εάν τελικά χρειάζεται για το συγκεκριμένο πρόβλημα.

Άρα θα χρησιμοποιηθεί αυτό το κριτήριο τερματισμού και ακολουθούν τα αποτελέσματα της εκπαίδευσης του βέλτιστου μοντέλου με την χρήση του **callback Early Stopping**.

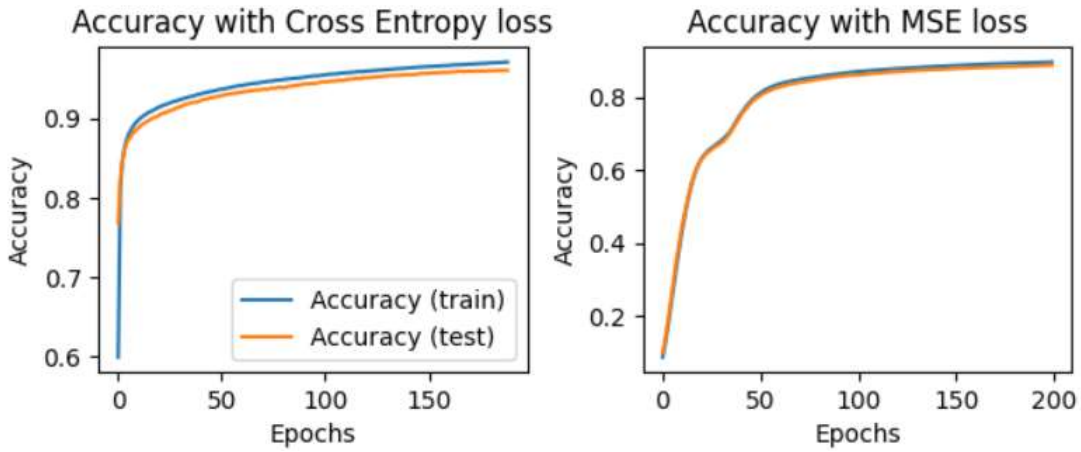
Θα χρησιμοποιηθεί **patience** ίσο με 5 και min_delta=0, δηλαδή αν δεν υπάρχει καθόλου βελτίωση για 5 συνεχόμενες εποχές, το μοντέλο μας θα σταματήσει να εκπαιδεύεται. Στο Early Stopping θα παρακολουθούνται οι αλλαγές στην ακρίβεια του validation set (val_accuracy). Χρησιμοποιούμε πολύ μικρό min_delta καθώς αυτό

αποτελεί την ελάχιστη αλλαγή που πρέπει να έχει η ακρίβεια ώστε να θεωρείται βελτίωση.

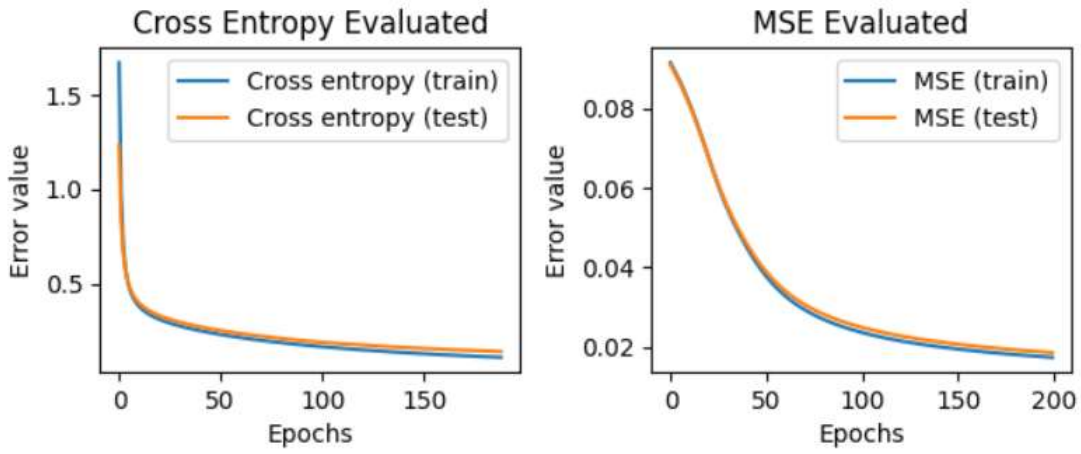
Σύγκριση μεταξύ μοντέλου με και χωρίς Early Stopping:

Αριθμός κρυφών επιπέδων	**CE loss**	**MSE**
1 κρυφό επίπεδο με 397 νευρώνες και Early Stopping	0.1243610	0.0166266
1 κρυφό επίπεδο με 397 νευρώνες χωρίς Early Stopping (200 εποχές)	0.1195953	0.0166156

Γραφικές με Early Stopping:



Εικόνα 19| Ακρίβεια με πρόωρο σταμάτημα



Εικόνα 20| CE και MSE loss με πρόωρο σταμάτημα

Με την χρήση του Early Stopping παρατηρείται πως με CE loss το μοντέλο μας εκπαιδεύεται μόνο για 189 εποχές, ενώ με MSE loss εκπαιδεύεται και για τις 200 εποχές. Η μετρική για το CE loss έχει χειρότερο αποτέλεσμα, κάτι το οποίο είναι λογικό καθώς το CE loss θα μίκραινε μέσα στις επόμενες 11 εποχές εκπαίδευσης. Επίσης, η ακρίβεια πέφτει επίσης, αλλά για 1%. Είναι πιθανό η ακρίβεια να βελτιωνόταν στις

επόμενες 11 εποχές, **οπότε δεν θα χρησιμοποιήσουμε τελικά το Early Stopping**, καθώς δεν βελτιώνει την εκπαίδευση του μοντέλου μας και δεν υπάρχει έτσι και αλλιώς το φαινόμενο του overfitting.

Α3. Μεταβολές στον ρυθμό εκπαίδευσης και σταθεράς ορμής

Αρχικά, θα αναφερθώ στην επίδραση των δύο παραμέτρων, **ρυθμός εκπαίδευσης και σταθερά ορμής**, στην εκπαίδευση και στην απόδοση ενός νευρωνικού δικτύου και μετά θα σχολιάσω τα αποτελέσματα των δοκιμών για τις διάφορες τιμές τους χρησιμοποιώντας το βέλτιστο μοντέλο που προέκυψε από το στ ερώτημα μαζί με το Early Stopping (γίνεται και χρήση 5fold cross validation όπως σε όλα τα ερωτήματα).

Ρυθμός εκπαίδευσης/μάθησης (Learning rate): Συμβολίζεται με η και χρησιμοποιείται για τον υπολογισμό των σφαλμάτων και την διόρθωση των συνδεσμικών βαρών των νευρώνων του δικτύου. Ο ρυθμός αυτός καθορίζει την ταχύτητα σύγκλισης της μάθησης και αφορά την ποσότητα κατά την οποία ενημερώνονται τα βάρη του νευρωνικού.

Αν έχουμε μεγάλο ρυθμό εκπαίδευσης, η σύγκλιση θα είναι γρηγορότερη, δεν θα έχουμε ομαλή τροχιά της καμπύλης των βαρών και θα βρισκόμαστε γύρω από τις βέλτιστες τιμές των βαρών. Επίσης, θα χρειάζονται και λιγότερες εποχές εκπαίδευσης καθώς θα είναι μεγαλύτερες οι αλλαγές στα βάρη.

Αν έχουμε μικρό ρυθμό εκπαίδευσης, η σύγκλιση θα είναι πιο αργή, θα έχουμε ομαλή τροχιά της καμπύλης των βαρών και μπορεί να εγκλωβιστούμε σε κάποιο τοπικό ελάχιστο. Επίσης, θα χρειάζονται και περισσότερες εποχές εκπαίδευσης καθώς θα είναι μικρότερες οι αλλαγές στα βάρη.

Για να εξαλείψουμε τα προβλήματα που προκύπτουν από τον μικρό και τον μεγάλο ρυθμό εκπαίδευσης, χρησιμοποιείται η σταθερά ορμής, η οποία τροποποιεί τον κανόνα αναπροσαρμογής των βαθών προσθέτοντας τον όρο α, θετικός αριθμός μεταξύ [0,1].

Σταθερά ορμής (Momentum): Συμβολίζεται με α και ισούται με την ποσότητα μεταβολής του βάρους από την προηγούμενο κύκλο εκπαίδευσης για τον καθορισμό της μεταβολής του στον επόμενο κύκλο. Αν αυξήσουμε πάρα πολύ τον ρυθμό εκπαίδευσης, τότε υπάρχει μεγάλη πιθανότητα η εκπαίδευση να γίνει ασταθής. Βοηθάει το μοντέλο ώστε να μην εγκλωβιστεί σε κάποιο τοπικό ελάχιστο χρησιμοποιώντας το «momentum» που έχει αποκτήσει το μοντέλο από τα προηγούμενα updates.Τροποποιημένος κανόνας δέλτα με ένταξη της ορμής: $\boldsymbol{\Delta w_{ji}(\eta) = a\Delta w_{ji}(n-1) + \eta\delta_j(n)y_1 \cdot (n)}$

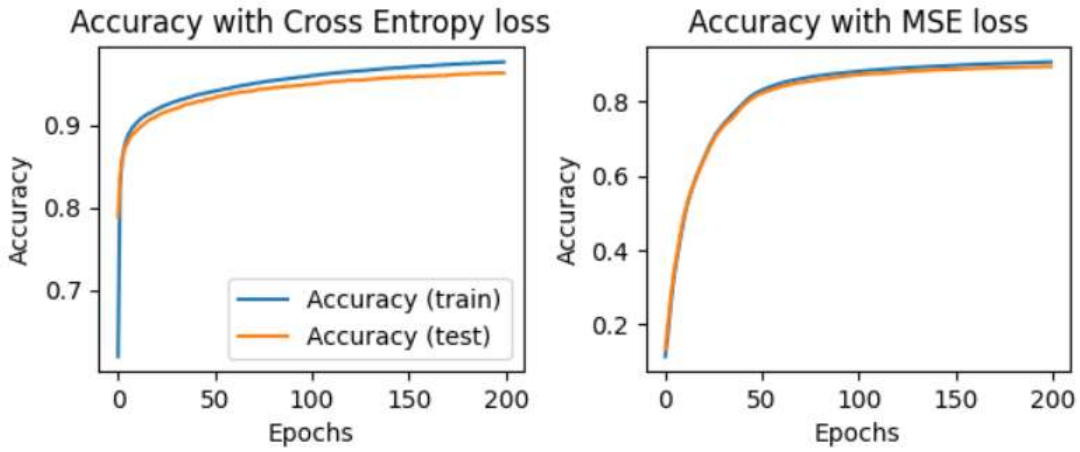
Το momentum είναι ο όρος α, και έχει τιμές μεταξύ 0 και 1. Έχει τιμή μεγαλύτερη ή ίση του μηδενός και μικρότερη της μονάδας, ώστε οι παλιότερες μεταβολές να επηρεάζουν λιγότερο το μοντέλο και έτσι να εξασφαλίζεται η σύγκλιση του

μοντέλου. Επιταχύνει ουσιαστικά το μοντέλο όταν βρίσκεται σε ρηχό (shallow) τοπικό ελάχιστο, ώστε τελικά να βρεθεί κάποιο σημαντικό ολικό ελάχιστο!

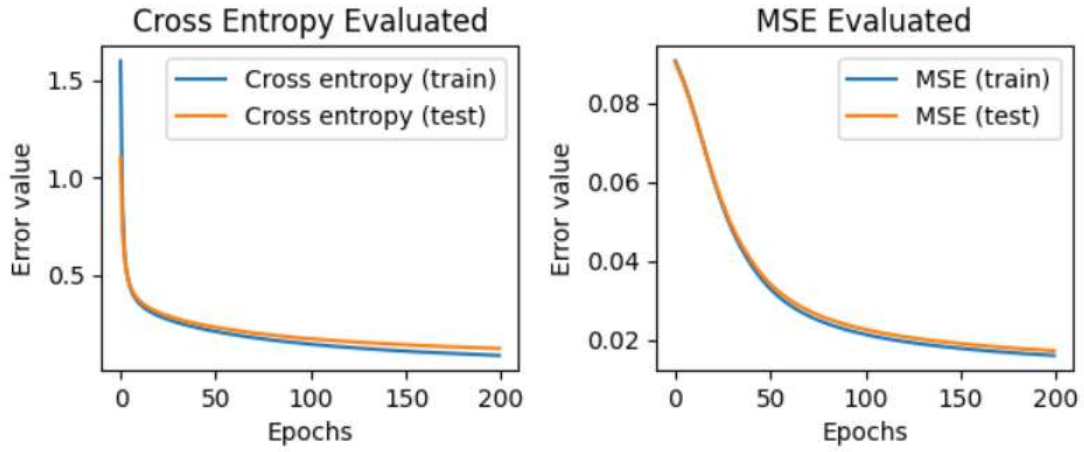
η	**m**	**CE loss**	**MSE**
0.001	0.2	0.10751774	0.01532420
0.001	0.6	0.07797802	0.01234485
0.05	0.6	0.08263752	0.00338775
0.1	0.6	0.08744687	0.00314386

η = 0.001 και m = 0.2 :

(96.7% ακρίβεια έναντι 90.7%)



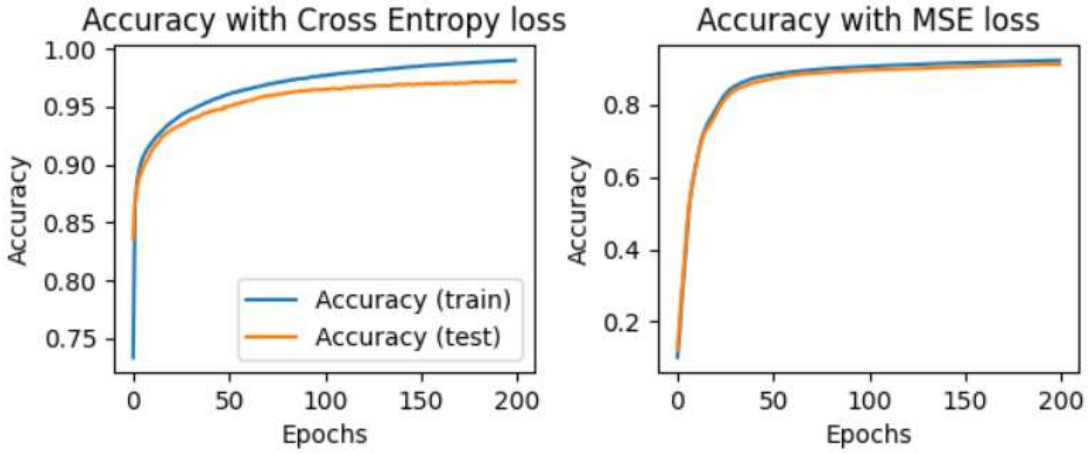
Εικόνα 21| Ακρίβεια για η=0.001 και m=0.2



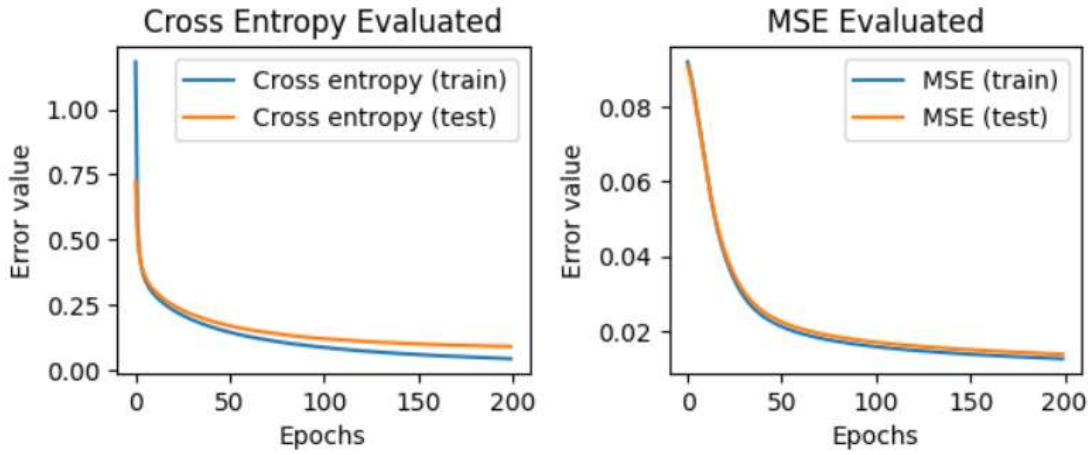
Εικόνα 22| CE και MSE loss για η=0.001 και m=0.2

η = 0.001 και m = 0.6:

(98% ακρίβεια έναντι 92.3%)



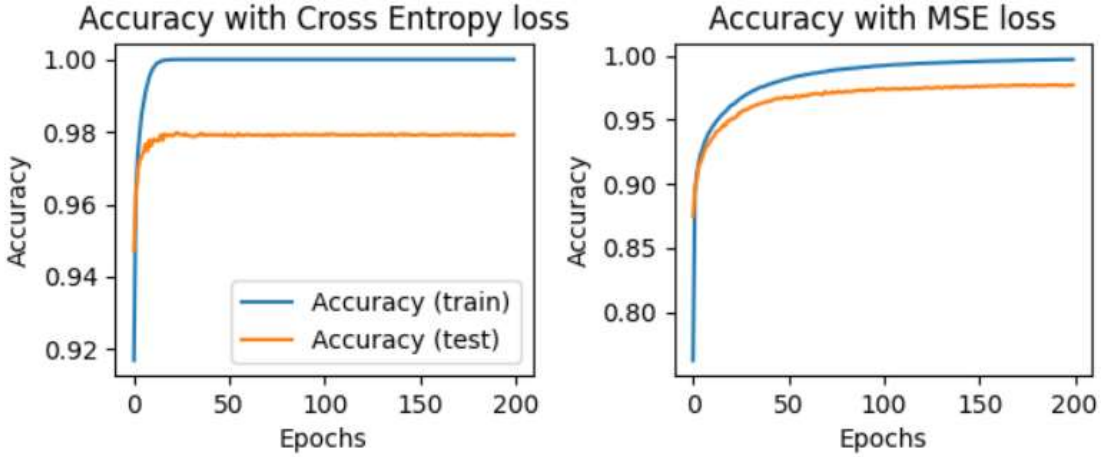
Εικόνα 23| Ακρίβεια για η=0.001 και m=0.6



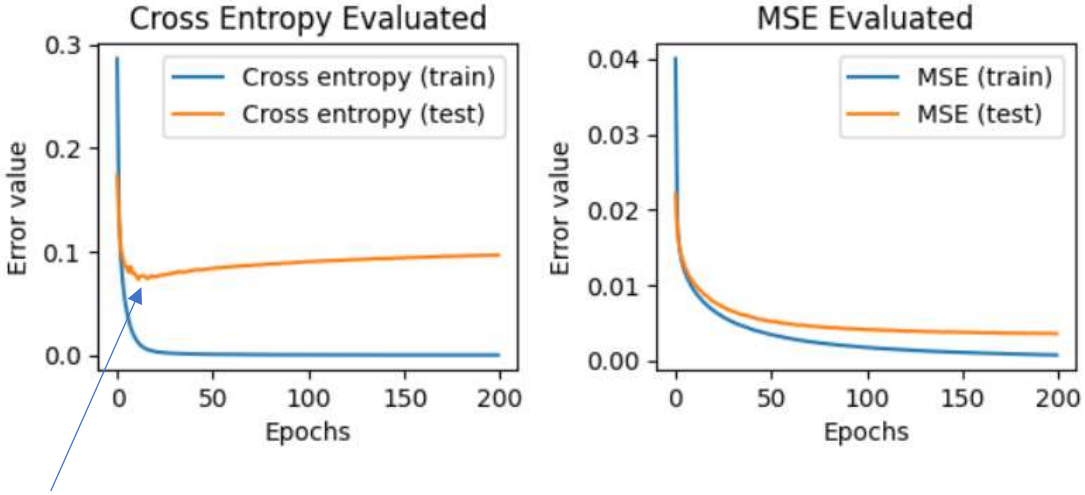
Εικόνα 24| CE και MSE loss για η=0.001 και m=0.6

η = 0.05 και m = 0.6:

(98.1 % ακρίβεια έναντι 97.8%)



Εικόνα 25| Ακρίβεια για η=0.05 και m=0.6

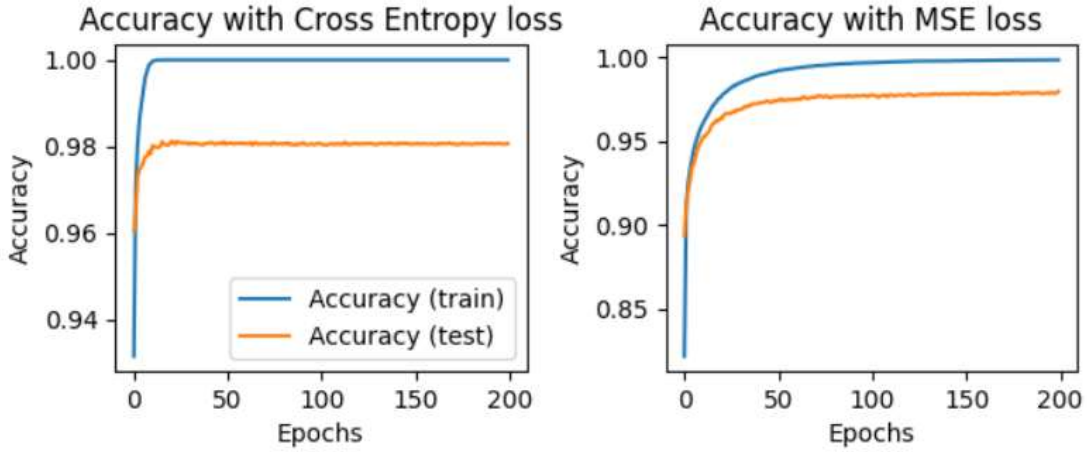


Εικόνα 26| CE και MSE loss για η=0.05 και m=0.6

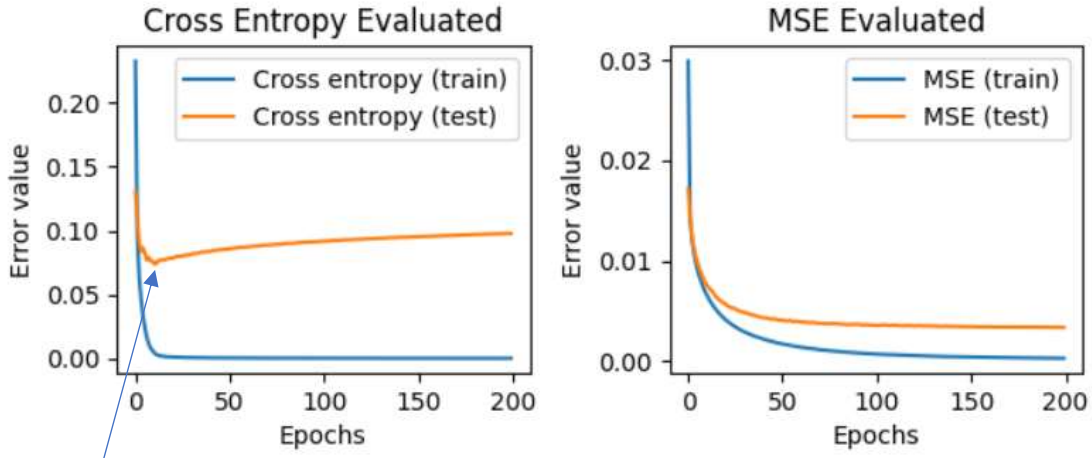
Απότομη και γρήγορη σύγκλιση

η = 0.1 και m = 0.6:

(98.2% ακρίβεια έναντι 97.9%)



Εικόνα 27| Ακρίβεια για η=0.1 και m=0.6



Εικόνα 28| CE και MSE loss για η=0.1 και m=0.6

Ακόμη πιο απότομη σύγκλιση

Αρχικά, όσο μεγαλώνει το learning rate, παρατηρούμε πολύ πιο γρήγορη σύγκλιση της εκπαίδευσης του μοντέλου καθώς μαθαίνει από τις πρώτες 7 εποχές καταφέρνει 99% train set accuracy και 97.6% validation accuracy. Με learning rate 0.001 και αυξημένο momentum (0.6) παρατηρούμε πιο γρήγορη σύγκλιση και βελτίωση των αποτελεσμάτων στην ακρίβεια και στις γραφικές των συναρτήσεων κόστους. Άρα, το momentum αποτελεί μία παράμετρο η οποία μπορεί να βοηθήσει αισθητά το μοντέλο μας στο πρόβλημα που εξετάζουμε.

Με 0.1 learning rate καταφέρνει 99% και 97.6 ακρίβεια στην $5^{η}$ εποχή, 2 εποχές νωρίτερα από ότι με 0.05 learning rate.

Αυτά τα νέα αποτελέσματα με τα μεγαλύτερα learning rates είναι δικαιολογημένα, καθώς όταν αυξάνεται αυτή η παράμετρος, τα «steps» που παίρνει το μοντέλο σε κάθε

update είναι μεγαλύτερα και έτσι το error μειώνεται πιο γρήγορα και η ακρίβεια αυξάνεται.

Με momentum 0.6 έχουμε πιο γρήγορη σύγκλιση και αυτό φαίνεται περισσότερο στις γραφικές του MSE loss. Επίσης, παρατηρούμε πως με learning rate 0.05 και 0.1, η ακρίβεια με MSE loss αυξάνεται παρά πολύ και γίνεται σχεδόν ίδια με την ακρίβεια με CE loss. Με αυτά τα learning rates παρατηρείται όμως και μία αύξηση στην CE loss μετά την σύγκλιση. Αυτό οφείλεται στο αυξημένο learning rate. Οι αλλαγές στα βάρη είναι πολύ πιο απότομες και οδηγούν την συνάρτηση κόστους στο να αποκλίνει (η συνάρτηση κόστους αποκλίνει από το ελάχιστο),

Θα χρησιμοποιήσουμε το μοντέλο με learning rate 0.05 και momentum 0.6, καθώς έχει πολύ καλά αποτελέσματα ειδικά στο MSE loss (καλύτερα από ότι χωρίς την χρήση momentum) και πολύ ομαλές γραφικές παραστάσεις. Βέβαια, παρατηρείται λίγο το φαινόμενο απόκλισης στο CE loss function, όμως θα προσπαθήσουμε να το αντιμετωπίσουμε στο επόμενο ερώτημα.

Α4. Ομαλοποίηση

Σε προηγούμενο ερώτημα μιλήσαμε για το Early Stopping κι την επίδραση του στην αποφυγή της υπερεκπαίδευσης του μοντέλου μας.

Σε αυτό το ερώτημα θα αναφερθούμε και στην **L2 ομαλοποίηση (L2 regularization)** και το πως βοηθάει και αυτή στην αποφυγή της υπερεκπαίδευσης και στην βελτίωση της γενικευτικής ικανότητας του μοντέλου μας.

L2 ομαλοποίηση: Καθώς εκπαιδεύουμε το μοντέλο μας, τα βάρη γίνονται όλο και περισσότερο εξειδικευμένα στα δεδομένα εκπαίδευσης, οδηγώντας έτσι στην υπερεκπαίδευση. Τα βάρη θα μεγαλώνουν ώστε να μπορούν να αντιμετωπίσουν τα samples των δεδομένων εκπαίδευσης. Όμως, μεγάλα βάρη οδηγούν σε ένα ασταθές μοντέλο, δηλαδή ακόμη και μία ελάχιστη αλλαγή στα δεδομένα εισόδου του θα οδηγεί σε μεγάλες αλλαγές στην έξοδο του. Εμείς επιθυμούμε να τιμωρούμε αυτά τα μεγάλα βάρη, κάτι το οποίο γίνεται με βάση το μέγεθος τους. Εδώ εισάγεται η L2, το οποίο αποτελεί τρόπο υπολογισμού του μεγέθους των βαρών.

Για την ακρίβεια, η L2 ασχολείται με την μείωση του αθροίσματος των τετραγώνων των τιμών των βαρών. Η L2 τιμωρεί πολύ αυστηρά τα μεγαλύτερα βάρη, όμως καταλήγει σε λιγότερο διάσπαρτα βάρη, σε αντίθεση με την L1. Η L2 χρησιμοποιείται πιο συχνά και αναφέρεται ως **weight decay**.

Η L2 ομαλοποίηση χρησιμοποιεί την παράμετρο alpha, η οποία αποτελεί την ποσότητα κατά την οποία θα τιμωρηθεί το μοντέλο με βάση το μέγεθος των βαρών. Παίρνει τιμές ανάμεσα στο 0 και στο 1. Αν είναι κοντά στο 1, το μοντέλο μπορεί να οδηγηθεί σε υποεκπαίδευση καθώς θα έχει υποτιμήσει τα βάρη του. Αν είναι κοντά στο 0, το μοντέλο θα οδηγηθεί σε υπερεκπαίδευση για τον αντίθετο λόγο.

Η L2 ομαλοποίηση ενθαρρύνει την χρήση όλων των εισόδων του νευρωνικού έστω και λίγο, σε αντίθεση με την L1 ομαλοποίηση που μπορεί να καταλήξει στην χρήση μόνο μερικών από τις σημαντικές εισόδους του νευρωνικού.

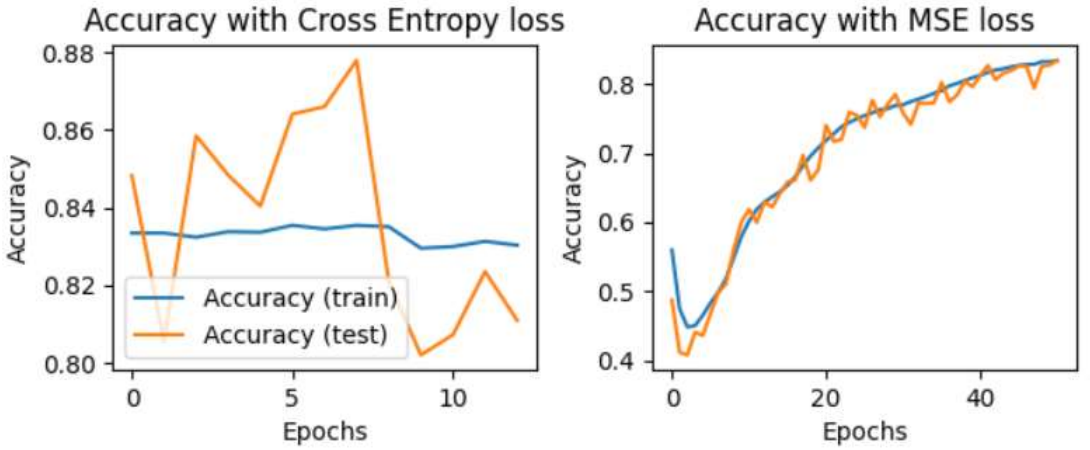
$$L(x,y) \equiv \sum_{i=1}^{n}(y_i - h_\theta(x_i))^2 + \lambda \sum_{i=1}^{n} \theta_i^2$$

Στα παρακάτω πειράματα χρησιμοποιείται learning rate=0.05 και momentum=0.6. Επειδή παρατηρήθηκε μία μικρή απόκλιση στην γραφική της CE loss στο προηγούμενο ερώτημα, αποφασίστηκε τελικά να χρησιμοποιηθεί Early Stopping με min_delta=0.001 σε αυτό το ερώτημα ώστε να βοηθήσει το μοντέλο και να φανεί τελικά η επίδραση του στα νευρωνικά δίκτυα.

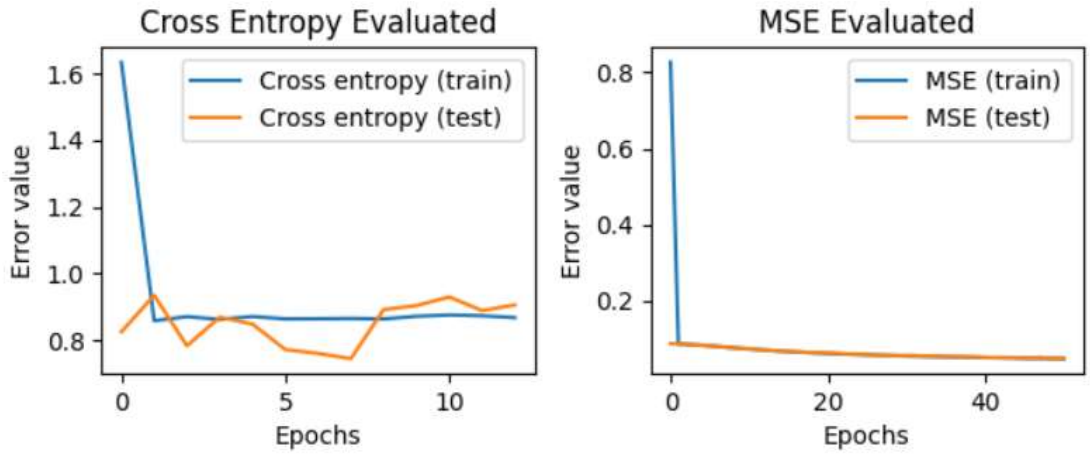
Συντελεστής φθοράς	**CE loss**	**MSE**
0.1	0.8725322	0.0466965
0.5	1.7506638	0.0896158
0.9	2.1321983	0.0898100

Alpha = 0.1:

(82.5% ακρίβεια έναντι 84.5%)



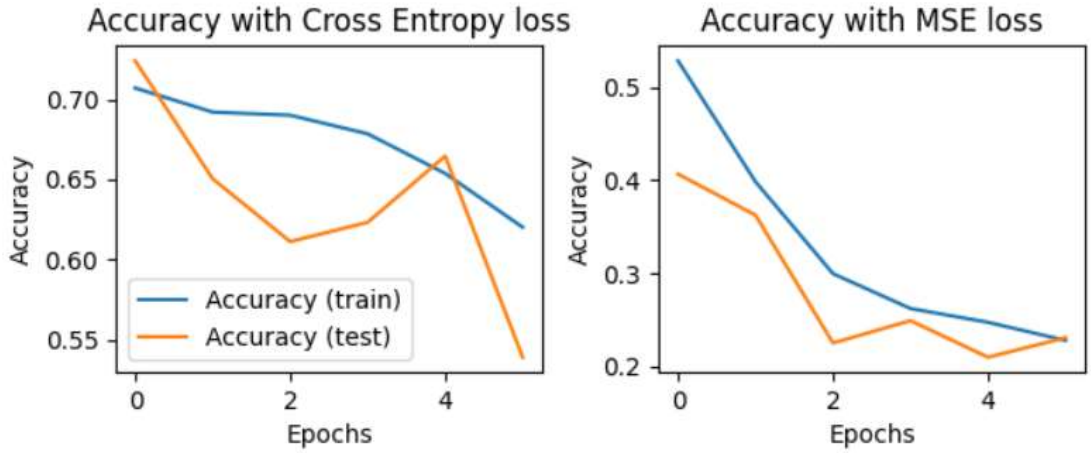
Εικόνα 29| Ακρίβεια για alpha=0.1



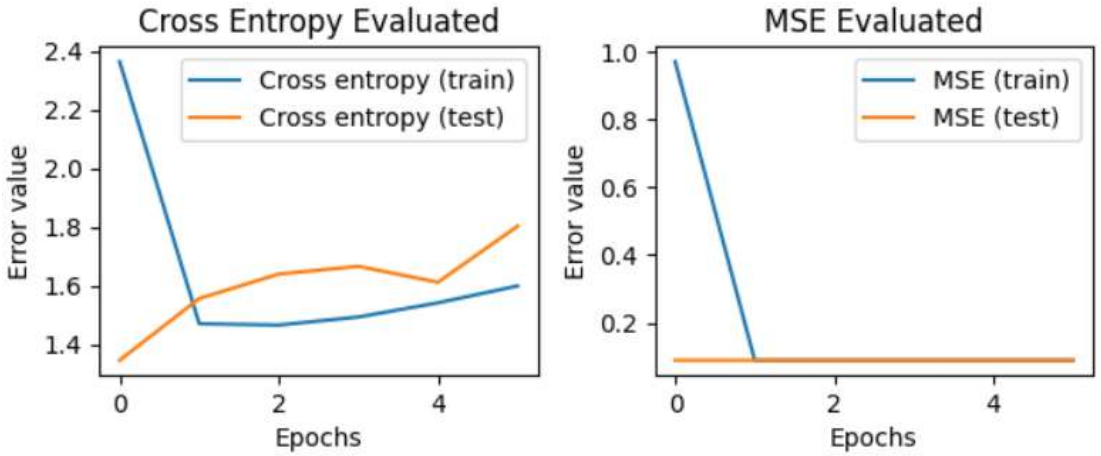
Εικόνα 30| CE και MSE loss για alpha=0.1

Alpha = 0.5:

(54% ακρίβεια έναντι 23%)



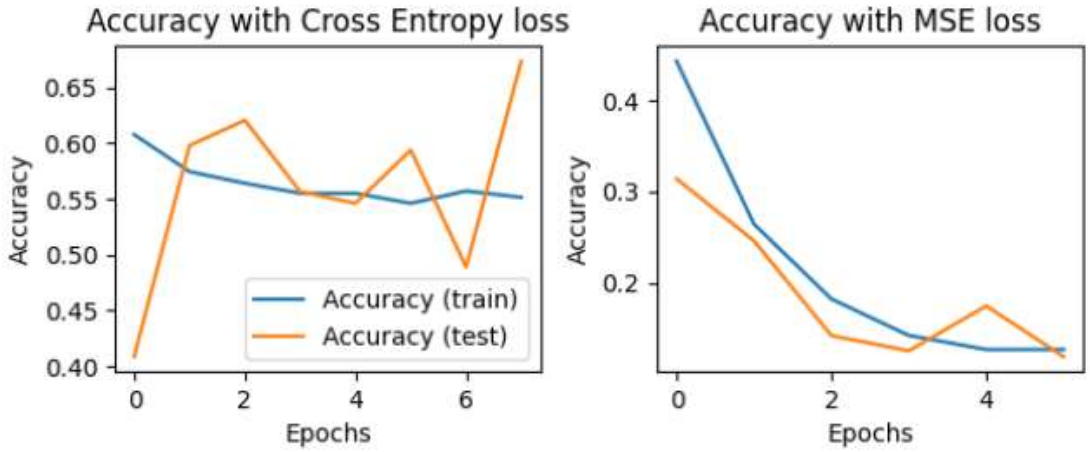
Εικόνα 31| Ακρίβεια για alpha=0.5



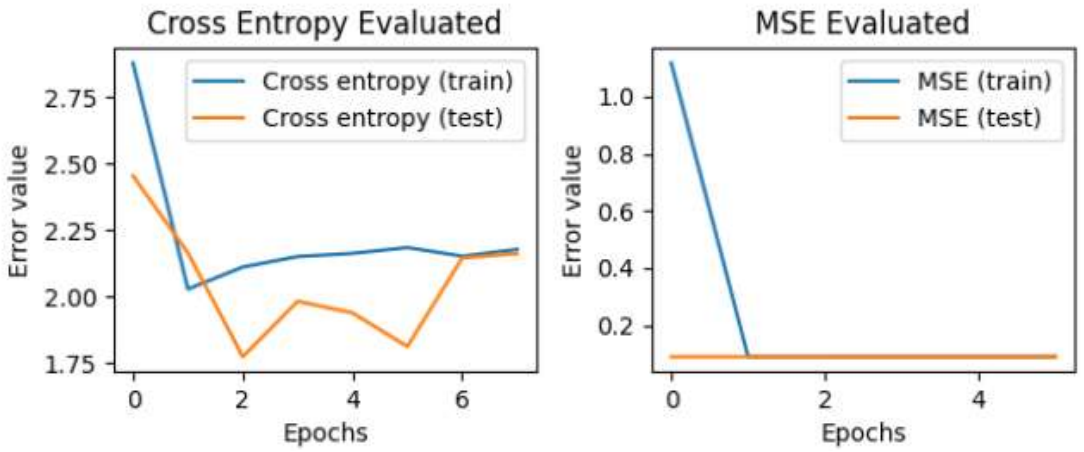
Εικόνα 32| CE και MSE loss για alpha=0.5

Alpha = 0.9:

(69% ακρίβεια έναντι 11.78%)



Εικόνα 33| Ακρίβεια με alpha=0.9



Εικόνα 34| CE και MSE loss για alpha=0.9

Το Early Stopping όντως βοήθησε στην εκπαίδευση του μοντέλου μας και επηρέασε πάρα πολύ τις εποχές εκπαίδευσης του.

Παρατηρείται πως όσο αυξάνεται η παράμετρος της L2 ομαλοποίησης, τα αποτελέσματα στην ακρίβεια και στις CE και MSE συναρτήσεις κόστους γίνονται χειρότερα. Αυτά τα αποτελέσματα ήταν αναμενόμενα καθώς η L2 ομαλοποίηση μειώνει την επίδραση κάποιων εισόδων και προσπαθεί να μειώσει την υπερεκπαίδευση του μοντέλου τιμωρώντας κάποια βάρη. Βέβαια, δεν παρατηρείται ποτέ υπερεκπαίδευση στα προηγούμενα πειράματα με το μοντέλο μας, οπότε η L2 δεν χρησιμοποιείται εδώ για την μείωση της υπερεκπαίδευσης αλλά ούτε και στην ικανότητα γενίκευσης του μοντέλου μας. Οι ταλαντώσεις (αυξομειώσεις) που κάνουν οι γραφικές παραστάσεις οφείλονται στην προσπάθεια της L2 να τιμωρήσει τα μεγάλα βάρη και να ομαλοποιήσει την εκπαίδευση του μοντέλου, αλλά δεν τα καταφέρνει ιδιαίτερα καλά.

Η ικανότητα γενίκευσης του μοντέλου ενισχύεται θεωρητικά με την χρήση της L2 ομαλοποίησης σε αντάλλαγμα με την μείωση της απόδοσης στις μετρικές που

αναλύουμε, όμως το μοντέλο μας ήδη γενίκευε σχεδόν τέλεια και η L2 ομαλοποίηση όχι μόνο δεν το βοήθησε, αλλά το επηρέασε αρνητικά. Το L2 γενικά παίρνει τιμές μέχρι το πολύ 0.1, οπότε ήταν αναμενόμενη η τόσο κακή απόδοση του μοντέλου μας με alpha 0.5 και 0.9. Πιθανότατα με alpha πχ 0.0001 να αντιδρούσε καλύτερα.

A5. Convolutional Neural Network

Στα προηγούμενα ερωτήματα παρουσιάστηκε η επίλυση του προβλήματος κατηγοριοποίησης των εικόνων του MNIST dataset με χρήση μοντέλων βαθιάς μάθησης μόνο με Dense layers.

Στο ερώτημα αυτό θα παρουσιαστεί η επίλυση του προβλήματος αυτού αλλά με χρήση Συνελικτικών Νευρωνικών Δικτύων! Θα παρουσιαστούν μερικά διαφορετικά τέτοια μοντέλα, με διαφορετικές αρχιτεκτονικές και θα αιτιολογηθεί η χρήση παραμέτρων και layers, καθώς και θα υπάρξει και σύγκριση μεταξύ των διαφορετικών αρχιτεκτονικών ώστε να καταλήξουμε στο καλύτερο μοντέλο για το πρόβλημα αυτό. **Επισημαίνεται πως δεν θα χρησιμοποιηθούν έτοιμα υλοποιήμενα και βελτιστοποιημένα CNN μοντέλα από το διαδίκτυο, αλλά θα γίνει προσπάθεια για κατασκευή δικών μου CNN μοντέλων με 1D layers.**

Ποια είναι η διαφορά όμως των Συνελικτικών Δικτύων από τα δίκτυα του προηγούμενου ερωτήματος;

Συνελικτικό Νευρωνικό Δίκτυο| Βασική διαφορά του με τα κλασσικά νευρωνικά δίκτυα είναι ότι δέχεται ως είσοδο έναν πολυδιάστατο πίνακα και όχι ένα διάνυσμα μιας διάστασης. Αυτό δίνει την δυνατότητα στο νευρωνικό να διατηρεί και να αξιοποιεί την τρισδιάστατη δομή των εικόνων, η οποία τρισδιάστατη δομή αλλάζει από επίπεδο σε επίπεδο, ενώ στα απλά νευρωνικά δίκτυα αλλάζει μόνο η μία διάσταση.

Convolutional Layer: Η συνέλιξη αποτελεί μία γραμμική πράξη που περιλαμβάνει πολλαπλασιασμό των βαρών με την είσοδο, παράγοντας έτσι μία έξοδο. Ο πολλαπλασιασμός αυτό γίνεται μεταξύ του διανύσματος εισόδου (τα δεδομένα μας) και ενός διανύσματος βαρών που ονομάζεται kernel ή αλλιώς φίλτρο (filter). Ουσιαστικά αποτελεί το άθροισμα του εσωτερικού γινομένου ανά element των δύο αυτών διανυσμάτων. Μέσω της εφαρμογής του kernel στα δεδομένα το layer αυτό εξάγει σημαντικά χαρακτηριστικά. Όσο περισσότερα Convolutional Layers έχουμε, τόσα περισσότερα διαφορετικά χαρακτηριστικά μπορούν να εξαχθούν από τα δεδομένα. Δημιουργούνται χάρτες χαρακτηριστικών μέσω της εφαρμογής των φίλτρων στην είσοδο.(feature maps).

Kernels (filters): Φίλτρο συνήθως περιττού μεγέθους το οποίο εφαρμόζεται σε κάθε element του διανύσματος εισόδου. Όσο μεγαλύτερο είναι το μέγεθος του kernel, τόσο μικρότερο είναι το διάνυσμα εξόδου του Conv layer.

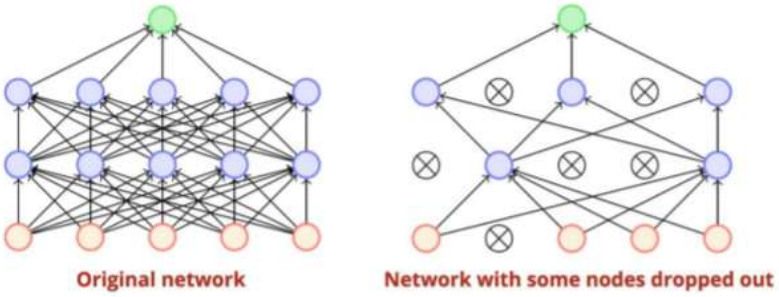
Strides: Το βήμα κατά το οποίο ολισθαίνει το φίλτρο (kernel). Χρησιμοποιήθηκε strides=1, δηλαδή ένα element την φορά, καθώς αποτελεί την πιο συχνή τιμή του (strides μεγαλύτερο του 1 μικραίνουν απότομα τους όγκους και χρειάζονται προσοχή). Εφόσον χρησιμοποιούμε pooling layer, δεν τίθεται τόσο θέμα η αύξηση των strides, καθώς το strides αυξάνεται όταν επιθυμούμε να μην

χρησιμοποιήσουμε pooling και να μειώσουμε τις χωρικές διαστάσεις, μειώνοντας έτσι και την πολυπλοκότητα του μοντέλου και να αυξήσουμε την ταχύτητα του.

Max Pooling Layer: Γενικά τα pooling layers συνοψίζουν τις εξόδους γειτονικών νευρώνων (που αποτελούν γκρουπς) εντός ενός παραθύρου με μία τιμή, η οποία τιμή μπορεί να αποτελεί το μέσο όρο ή το μέγιστο των επιμέρους τιμών των νευρώνων. Το pooling ουσιαστικά αποτελεί μέθοδος υπο-δειγματοληψίας των δεδομένων που μειώνει αρκετά τα δεδομένα και τις πράξεις που πρέπει να γίνουν (μειώνει το μέγεθος του feature map) . Θα χρησιμοποιηθεί η μέγιστη για τιμή ανά δύο elements, δηλαδή ανά δύο elements θα υπολογίζει το μέγιστο τους και θα τα αντικαθιστά με αυτή την τιμή.

Ο συνδυασμός Convolutional και Pooling layers οδηγεί στην δυνατότητα συνδυασμού των τοπικών χαρακτηριστικών ώστε να μάθει το δίκτυο περισσότερα ολικά χαρακτηριστικά (global) των δεδομένων.

Dropout Layer: Layer που χρησιμοποιείται για την αντιμετώπιση της υπερεκπαίδευσης στο μοντέλο και αύξηση της γενικευτικής ικανότητας, είναι μέθοδος ομαλοποίησης. Ουσιαστικά απενεργοποιεί τυχαία νευρώνες με πιθανότητα που την ορίζει ο χρήστης (συνήθως 0.25 στα Convolutional Layers και 0.5 στα Dense layers). Οι νευρώνες αυτοί δεν θα συνεισφέρουν καθόλου στην εμπρός διάδοση των σημάτων εκπαίδευσης και ούτε στο back-propagation. Έτσι οι διπλανοί νευρώνες αναγκάζονται να εξειδικευτούν καλύτερα με αποτέλεσμα το μοντέλο να καταφέρνει να μάθει πιο εύκολα κάποια πιο ισχυρά χαρακτηριστικά.



Εικόνα 35| Dropout

Flatten: Μετατρέπει τον τελικό χάρτη χαρακτηριστικών (feature map) σε ένα 1D διάνυσμα. Ουσιαστικά συνδυάζει όλα τα τοπικά χαρακτηριστικά που βρήκαν πριν τα convolutional layers.

Fully Connected Layer (Dense layers): Έπειτα από το Flatten, έρχεται layer το οποίο έχει πλήρως συνδεδεμένους όλους τους νευρώνες του με το προηγούμενο επίπεδο. Ουσιαστικά αποτελεί ένα ΤΝΔ όπως στα πρώτα ερωτήματα. Στα επίπεδα αυτά μαθαίνονται οι μη γραμμικοί συνδυασμοί των σημαντικών χαρακτηριστικών που εξήγαγε το δίκτυο στα προηγούμενα επίπεδα.

Θα χρησιμοποιηθεί **ReLU** για όλα τα layers εκτός του τελευταίου Dense layers στο οποίο θα χρησιμοποιηθεί **softmax**, αφού έχουμε πρόβλημα κατηγοριοποίησης 10

κλάσεων. Επίσης, θα χρησιμοποιηθεί και Categorical Cross Entropy και MSE σαν συναρτήσεις κόστους όπως στα προηγούμενα ερωτήματα (για να γίνουν οι συγκρίσεις) για τους ίδιους λόγους και θα χρησιμοποιηθεί και **SGD** για optimizer.

Θα χρησιμοποιηθεί **patience** ίσο με 5 και min_delta=0, δηλαδή αν δεν υπάρχει καθόλου βελτίωση για 5 συνεχόμενες εποχές, το μοντέλο μας θα σταματήσει να εκπαιδεύεται. Στο Early Stopping θα παρακολουθούνται οι αλλαγές στην ακρίβεια του validation set (val_accuracy). Το min_delta αποτελεί την ελάχιστη αλλαγή που πρέπει να έχει η ακρίβεια ώστε να θεωρείται βελτίωση.

(Τα layers που αναφέρονται είναι 1D)

Αρχιτεκτονικές CNN

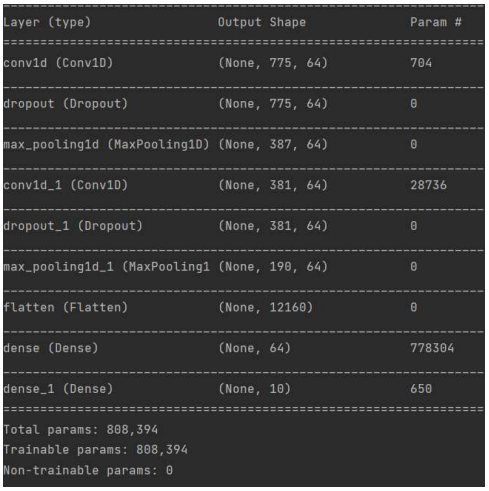
Ως πρώτη αρχιτεκτονική θα χρησιμοποιηθεί απλά ένα Convolutional layer με 64 filters μεγέθους 10 και strides 1, ακολουθόμενο από ένα dropout και ένα Average Pooling layer και τέλος 2 Dense layers 64 και 10 νευρώνων αντίστοιχα. Η αρχιτεκτονική αυτή αποτελεί μία από τις πιο απλές που μπορούσε να λειτουργήσει για το πρόβλημα αυτό. Παρατηρείται πως ακόμα και με ένα απλό CNN δίκτυο τα αποτελέσματα των μετρικών είναι καλύτερα από το καλύτερο μοντέλο που βρήκαμε στα πρώτα ερωτήματα. Η ακρίβεια όμως πέφτει λίγο, ειδικά στο train set. Το δίκτυο αυτό χρειάζεται λιγότερες εποχές εκπαίδευσης και δεν υπερεκπαιδεύεται.

Layer (type)	Output Shape	Param #
conv1d (Conv1D)	(None, 780, 64)	384
dropout (Dropout)	(None, 780, 64)	0
max_pooling1d (MaxPooling1D)	(None, 390, 64)	0
flatten (Flatten)	(None, 24960)	0
dense (Dense)	(None, 64)	1597504
dense_1 (Dense)	(None, 10)	650

Total params: 1,598,538
Trainable params: 1,598,538

Εικόνα 36| 1η CNN αρχιτεκτονική

Ως δεύτερη αρχιτεκτονική θα χρησιμοποιηθούν 2 Convolutional layers. Το 1^{ο} με 64 filters μεγέθους 10 και το δεύτερο με 64 filters μεγέθους 7, ακολουθόμενα από Dropout και Average Pooling layers και τέλος 2 Dense layers 64 και 10 νευρώνων αντίστοιχα. Επιλέχθηκαν αυτοί οι αριθμοί φίλτρων ώστε να εξεταστεί η χρήση ίδιου αριθμού φίλτρων σε συνεχόμενα convolutional layers για το πρόβλημα που εξετάζεται.

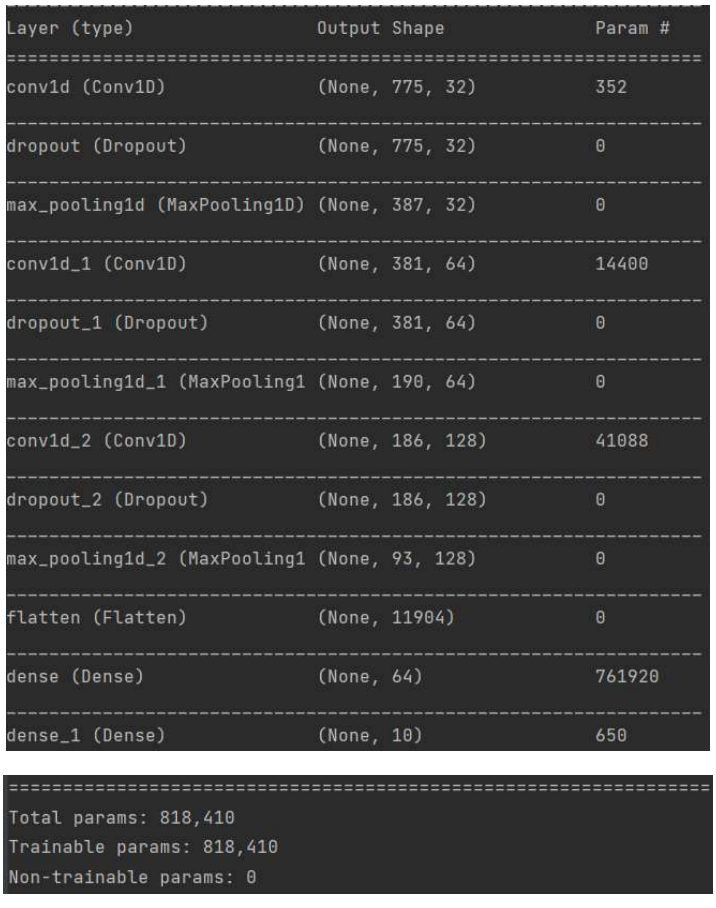


Layer (type)	Output Shape	Param #
conv1d (Conv1D)	(None, 775, 64)	704
dropout (Dropout)	(None, 775, 64)	0
max_pooling1d (MaxPooling1D)	(None, 387, 64)	0
conv1d_1 (Conv1D)	(None, 381, 64)	28736
dropout_1 (Dropout)	(None, 381, 64)	0
max_pooling1d_1 (MaxPooling1	(None, 190, 64)	0
flatten (Flatten)	(None, 12160)	0
dense (Dense)	(None, 64)	778304
dense_1 (Dense)	(None, 10)	650

Total params: 808,394
Trainable params: 808,394
Non-trainable params: 0

Εικόνα 37| 2η CNN αρχιτεκτονική

Ως τρίτη αρχιτεκτονική προσθέσαμε άλλα δύο επίπεδα, ένα ακόμη Convolutional Layer και ένα ακόμη Dense Layer. Σε αυτή την αρχιτεκτονική δοκιμάστηκε η αύξηση του αριθμού των filters από convolutional layer σε convolutional layer. Για την ακρίβεια, ξεκινήσαμε από 32 filters και φτάσαμε στα 128 filters. Στα Dense layers ξεκινήσαμε από τα 128 filters και φτάσαμε στα 64. Το τελευταίο layer όπως πάντα αντιπροσωπεύει την έξοδο. Επιλέχθηκαν αυτοί οι αριθμοί φίλτρων ώστε να εξεταστεί η χρήση αυξανόμενου αριθμού φίλτρων σε συνεχόμενα convolutional layers για το πρόβλημα που εξετάζεται.



Layer (type)	Output Shape	Param #
conv1d (Conv1D)	(None, 775, 32)	352
dropout (Dropout)	(None, 775, 32)	0
max_pooling1d (MaxPooling1D)	(None, 387, 32)	0
conv1d_1 (Conv1D)	(None, 381, 64)	14400
dropout_1 (Dropout)	(None, 381, 64)	0
max_pooling1d_1 (MaxPooling1	(None, 190, 64)	0
conv1d_2 (Conv1D)	(None, 186, 128)	41088
dropout_2 (Dropout)	(None, 186, 128)	0
max_pooling1d_2 (MaxPooling1	(None, 93, 128)	0
flatten (Flatten)	(None, 11904)	0
dense (Dense)	(None, 64)	761920
dense_1 (Dense)	(None, 10)	650

Total params: 818,410
Trainable params: 818,410
Non-trainable params: 0

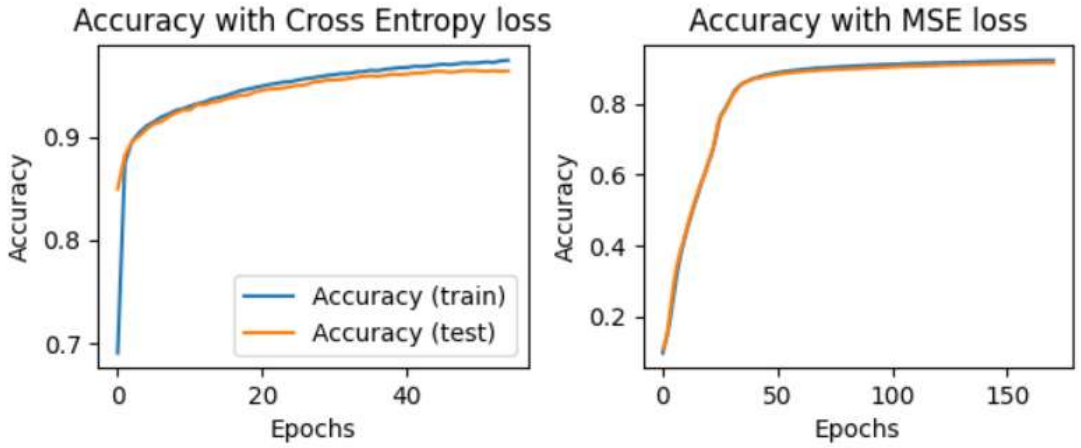
Εικόνα 38| 3η CNN αρχιτεκτονική

CNN	CE loss	MSE
1η αρχιτεκτονική	0.11470239	0.01237953
2η αρχιτεκτονική	0.10139454	0.01225871
3η αρχιτεκτονική	0.14218541	0.01155412

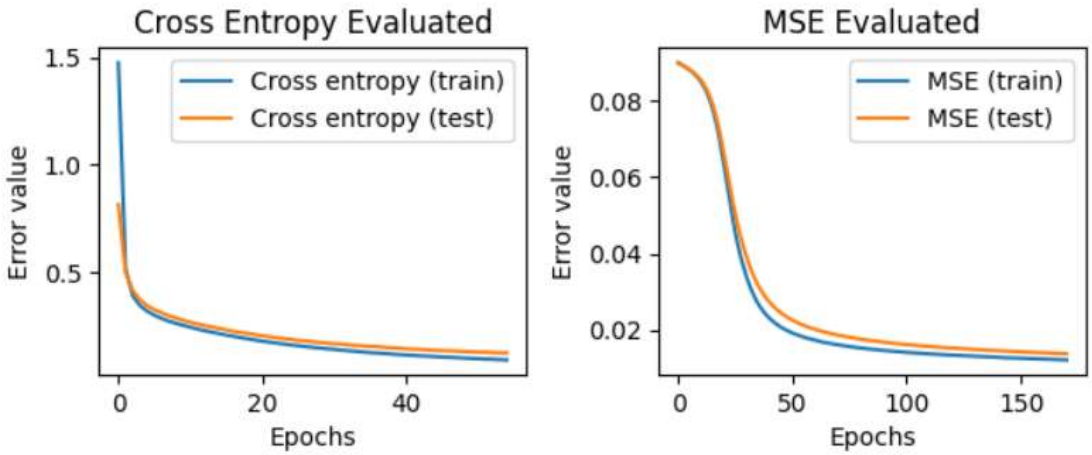
Γραφικές παραστάσεις πειραμάτων:

1η αρχιτεκτονική:

(96.7% ακρίβεια σε 53 εποχές έναντι 92.4% σε 185 εποχές)



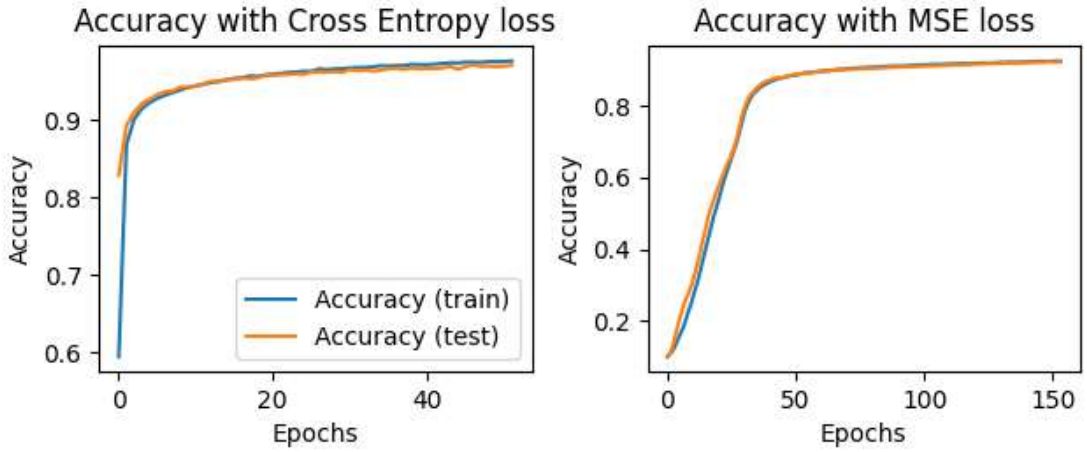
Εικόνα 39| Ακρίβεια για την 1η CNN αρχιτεκτονική



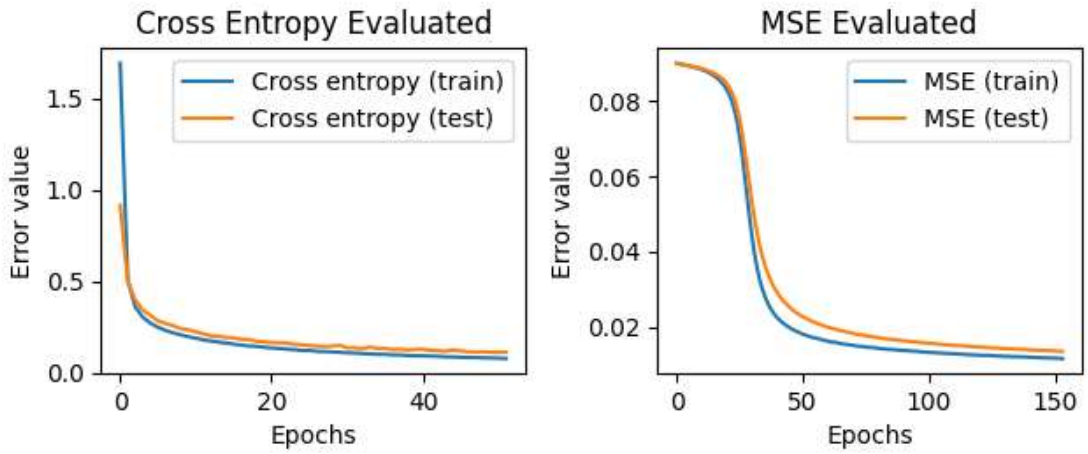
Εικόνα 40| CE και MSE loss για την 1η αρχιτεκτονική

2η αρχιτεκτονική:

(97.4% ακρίβεια σε 51 εποχές έναντι 93.5% σε 154 εποχές)



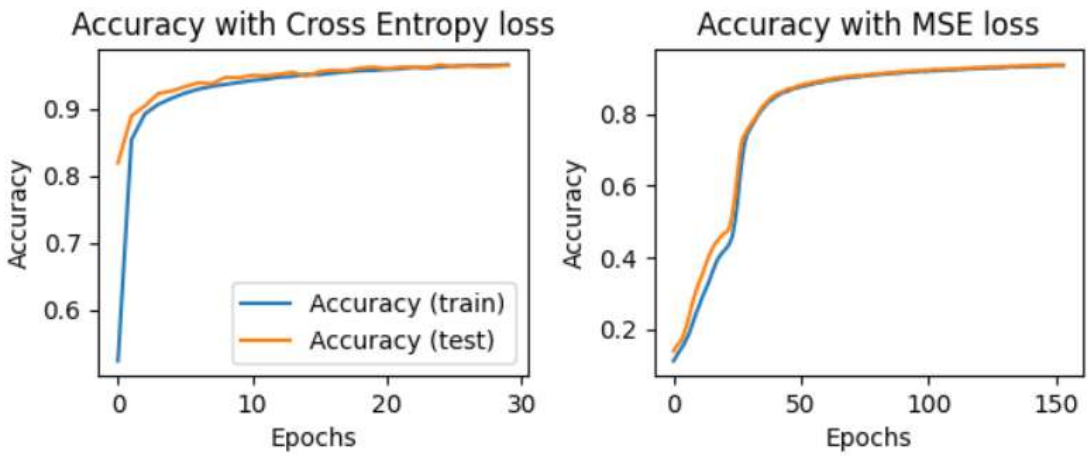
Εικόνα 41| Ακρίβεια για την 2η CNN αρχιτεκτονική



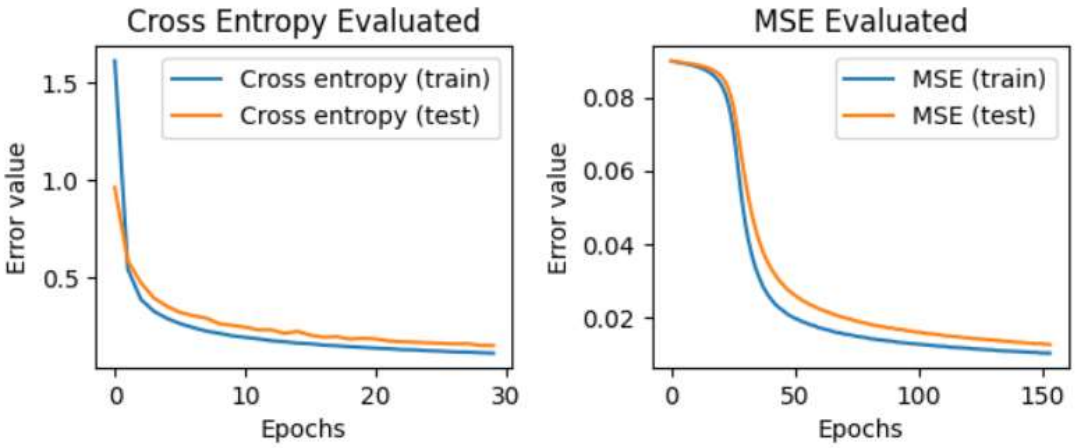
Εικόνα 42| CE και MSE loss για την 2η CNN αρχιτεκτονική

3η αρχιτεκτονική:

(96.6% ακρίβεια σε 30 εποχές έναντι 94% σε 154 εποχές)



Εικόνα 43| Ακρίβεια για την 3η CNN αρχιτεκτονική



Εικόνα 44| CE και MSE loss για την 3η CNN αρχιτεκτονική

Όσο αυξάνονται τα convolutional layers, παρατηρείται πως ειδικά η εκπαίδευση με CE loss τερματίζεται πολύ πιο γρήγορα. Αυτό σημαίνει πως το πιο περίπλοκο μοντέλο μαθαίνει πιο γρήγορα τα μοτίβα του προβλήματος, ταυτόχρονα όμως παίρνει πολύ περισσότερο χρόνο και υπολογιστικούς πόρους για να εκπαιδευτεί.

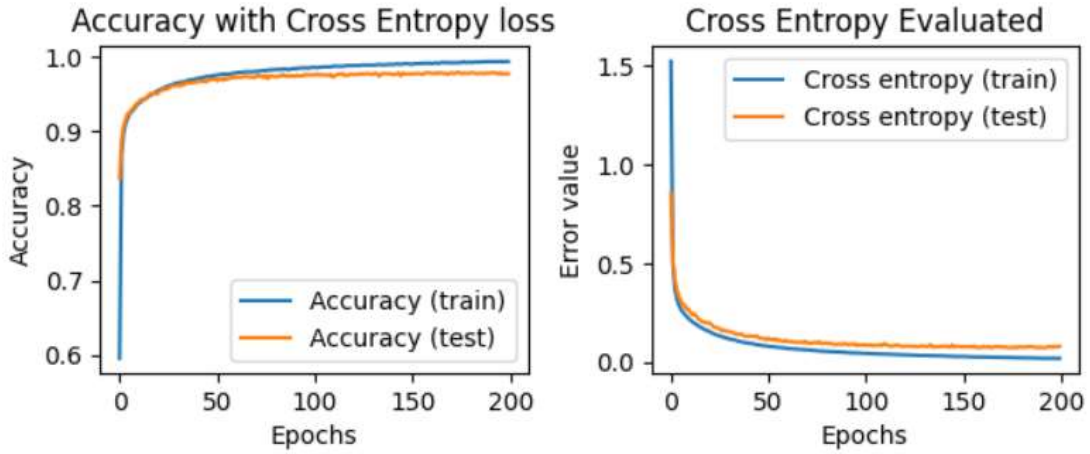
Τελικά, παρατηρούμε ότι τα καλύτερα αποτελέσματα για τις 2 μετρικές τα προσφέρει η δεύτερη αρχιτεκτονική, η οποία έχει και τις λιγότερες training παραμέτρους. Η τρίτη αρχιτεκτονική έχει ένα επιπλέον σετ Convolutional, Pooling, Dropout layers και είναι πιο περίπλοκο. Καταφέρνει περίπου την ίδια ακρίβεια σε λιγότερες εποχές, αλλά στο πρόβλημα μας δεν χρειάζεται ένα τόσο περίπλοκο δίκτυο με τόσες πράξεις να συμβαίνουν.

Η 2η αρχιτεκτονική ακολουθεί το μοτίβο του ίδιου αριθμού φίλτρων σε συνεχόμενα convolutional layers, οπότε θεωρούμε πως στο συγκεκριμένο πρόβλημα αυτό το μοτίβο αποδίδει καλύτερα.

Σύγκριση καλύτερου CNN με καλύτερο ΤΝΔ:

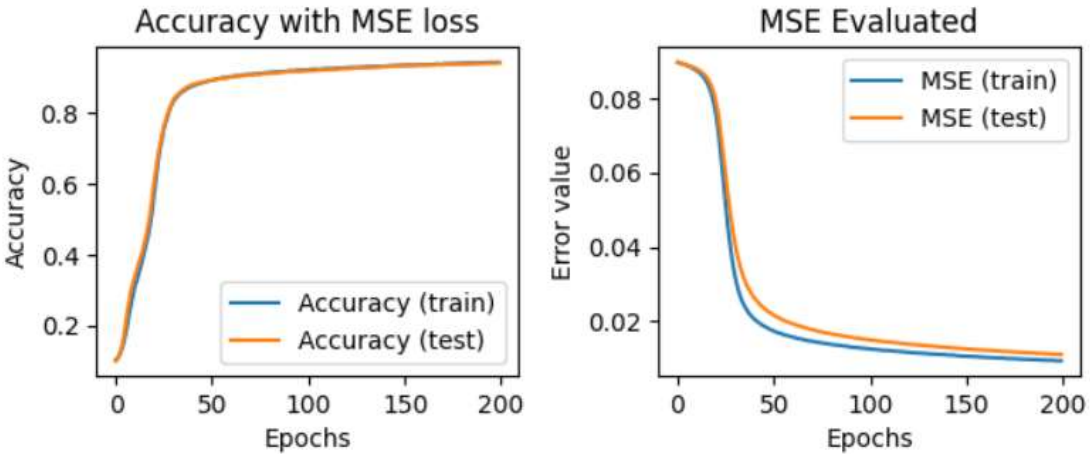
Η CNN αρχιτεκτονική μας με EarlyStopping callback καταφέρνει πολύ καλά αποτελέσματα και ας τελειώνει σε 51 εποχές με CE loss και σε 154 για MSE loss. Για να έχουμε μια πιο ολοκληρωμένη και σωστή σύγκριση CNN με ΤΝΔ, αφαιρέσαμε το EarlyStopping ώστε να τρέξει για 200 εποχές και το CNN, ώστε να συγκριθούν τα δύο νευρωνικά μεταξύ τους και για τις 200 εποχές. Με 200 εποχές εκπαίδευσης, το CNN κατάφερε πολύ καλύτερα αποτελέσματα και στην ακρίβεια και στις συναρτήσεις κόστους!

Σχεδόν 100% ακρίβεια για train set και 98% για test set με CE loss!



Εικόνα 45| Ακρίβεια και CE loss για 200 εποχές στο CNN

94.7% ακρίβεια για MSE loss!



Εικόνα 46| Ακρίβεια και MSE loss για 200 εποχές στο CNN

Σύγκριση καλύτερου CNN με καλύτερο ΤΝΔ	**CE loss**	**MSE**
ΤΝΔ	0.07797802	0.01234485
CNN με Early Stopping	0.10139454	0.01225871
CNN 200 epochs	0.06782519	0.00983888

Ο χρόνος βέβαια που χρειάζεται να ολοκληρωθεί η εκπαίδευση του CNN είναι μεγαλύτερος από του ΤΝΔ, καθώς είναι αρκετά πιο περίπλοκο δίκτυο με περισσότερα layers, δηλαδή ο χρόνος ολοκλήρωσης κάθε εποχής είναι αρκετά μεγαλύτερος. Πάντως, ακόμη και με CNN φαίνεται η επικράτηση της CE loss ως καλύτερης συνάρτησης κόστους από την MSE για το πρόβλημα κατηγοριοποίησης μας.